গ্নু-লিনাক্স ইশকুল

জিএলটি মধ্যমগ্রাম
glt-mad@ilug-cal.org

আজকের আলোচনা ব্যাশ শেল নিয়ে। প্রথমে তার কনফিগারেশন, সেখান থেকে ব্যাশের কাজ করার রকম নিয়ে কিছু কথা। কিছুটা আলোচনা, একটা ছবি সহ, শেল কী ভাবে আপনার আদেশ পালন করে, আপনার এবং কারনেলের ভিতর সংযোগ রাখে, আমরা করেছিলাম পাঁচ নম্বর দিনে। তখন সিস্টেমের বহু খুঁটিনাটি আমরা জানতাম না, সেই আলোচনাটাকেই এবার আমরা বাড়িয়ে নিয়ে যাব। কিছু কিছু শেল স্ক্রিপ্ট এবং শেলকে দিয়ে কী করে অনেকটা বা অনেক রকম কাজ একই সঙ্গে করা যায় তাই নিয়ে কিছু প্রসঙ্গ আমরা সাত এবং নয় নম্বর দিনেও এনেছি, সেই আলোচনাগুলোকে এবার আমরা আরো অনেকটা অব্দি নিয়ে যাব। আপনার সিস্টেমের ভিতরেই প্রচুর ডকুমেন্টেশন আছে ব্যাশ নিয়ে, তারপরেও আছে লিনাক্স ডকুমেন্টেশন প্রোজেক্ট মানে 'www.tldp.org' বা আছে ফ্যাক অর্গানাইজেশন মানে 'www.faq.org'। এবং এছাড়াও আরো প্রচুর প্রচুর ডকুমেন্টেশন ছড়িয়ে আছে নেটে। ব্যাশ শেল বা ব্যাশ স্ক্রিপ্ট সার্চে দিয়ে যদি একবারো গুগলি মেরে, 'www.google.com', শখানেকের কম লিংক পান, ফ্রিতে আমায় কেলিয়ে যেতে পারেন। এরকম প্রচুর মালপত্তর আমার কাছে নামানো আছে, আর আমি মানুষ হিশেবে অত্যন্ত উদার হওয়ায় আপনি যদি আমার সঙ্গে বসে সেসব পড়তে চান, আমার কোনো আপত্তি নেই। তবে মনে রাখবেন, নিখিল-ভারত-আইনপন্থী-হায়ার-কমিটির চেয়ারম্যান পদ পাওয়ার কথা চলছে আমার, তাই কোনো কপিরাইটরিক্ত জিনিষপত্তর আমার কাছ থেকে সিডিতে লিখে নেওয়ার আশা করবেন না। এমনকি আমার বার্নার ড্রাইভটা অব্দি কোনো বেআইনি জিনিষ কপি করতে দিলে কাজ থামিয়ে জনগনমন গাইতে শুরু করে। সোজা হয়ে দাঁড়িয়েও নেয়, রবীন্দ্রপ্রণামের মূল উদ্দিষ্ট ব্রিটিশ সম্রাট পঞ্চম জর্জ থেকে আজকের ভারতীয় সংবিধান অব্দি সকলের প্রতি বিনম্র শ্রদ্ধায় — মানে, শ্রদ্ধয়া দেয়ম, শ্রাদ্ধ পালন করে।

।। দিন দশ।।

১।। ব্যবহারকারীর প্রোগ্রামের কনফিগারেশন ফাইল

সাত নম্বর দিনের ২.৩ সেকশনে ব্যাশের কাছে তুলে রাখা আপনার আগে-ব্যবহৃত আদেশের ইতিহাসের প্রসঙ্গে আমরা এই আলোচনাটাকে একটু এনেছিলাম। আমরা একটা হোম ডিরেক্টরির মোট ডটানন ফাইল এবং আরসি (rc) ফাইলের একটা অসম্পূর্ণ তালিকা দিয়েছিলাম। এবার সেই ফাইলগুলোকে একটু বোঝার চেষ্টা করা যাক। তার আগে জানা দরকার, ব্যবহারকারীর প্রোগ্রাম বলতে আমরা কী বুঝছি। বাইনারি এবং সিস্টেম বাইনারির আলোচনায় আমরা রুটের সঙ্গে অন্য ব্যবহারকারীর অধিকারের তফাতের কথা বলেছিলাম। এছাড়া গ্লোবাল এবং লোকাল সেটিং-এর প্রসঙ্গটাও এখন আমাদের অপরিচিত নয়। খুব ছোট অর্থে আমরা লোকাল বলতে ধরব একজন নির্দিষ্ট ব্যবহারকারীর বেলায় যা সত্যি, কিন্তু সিস্টেমের যে কোনো ব্যবহারকারীর বেলায় নয়। ধরুন আমি একজন সাধারণ ব্যবহারকারী হিশেবে আমি এমন ব্যবস্থা করতেই পারি, সত্যিই এই রকম ব্যবস্থা করা আছে যে, আমি লগ-ইন করা মাত্র আমার শেল আমায় একটা ছোট্ট দেখে আপত্তিকর মানে অফেনসিভ বাণী শোনায় ফরচুনের। অফেনসিভ ফরচুন মানে কতটা অফেনসিভ সেটা সুরুচি এবং কালচারের সমস্ত পিছুটান থেকে সম্পূর্ণ মুক্ত না-হলে সত্যিই দেখতে যাওয়ার রিস্ক নেওয়া উচিত নয়, আমার বন্ধুরা জানেন যে গর্ব আমি অনায়াসে করতে পারি। বেশিরভাব ডিস্ট্রোই অফেনসিভ ফরচুন নিজের মধ্যে দেয়না, আলাদা করে ডাউনলোড করে ইনস্টল করে নিতে হয়। এবার দেখুন, এই '`fortune -os`' ব্যবস্থাটা, মানে সেই সব ফরচুন বাণী যারা শর্ট এবং অফেনসিভ, ছোট্ট এবং আপত্তিকর, শোনানোর ব্যবস্থাটা আমার মানে ইউজার 'dd'-র নিজের হোম ডিরেক্টরিতে করা আছে আমার ব্যাশের কনফিগারেশনে। কিন্তু এটা ইউজার 'manu' বা 'piu' বা 'atithi'-র কনফিগারেশনে করা নেই। তাই এটা সেই অর্থে লোকাল সেটিং। এরকম ইউজার প্রোগ্রামেও করা আছে। আমি ইম্যাক্স খুললেই সেটা আমার নিজের কিছু পছন্দ মোতাবেক চলে, সেটাও লোকাল সেটিং। এরকম অনেক প্রোগ্রামেই আছে। আবার ধরুন, আমি সেদিন ভিম (vim) সফটওয়ারটার একটা নতুন ভার্শন, ভার্শন ৬.২, যেটা সুজে ৮.২-এ দেওয়া ছিলনা, নামিয়েছিলাম নেট থেকে, 'vim-6.2.tar.bz2' ফাইলে এবং সেই কোঁকড়ানো সিন্দুকটাকে '/opt' ডিরেক্টরিতে রেখে '`tar xvjf vim-6.2.tar.bz2`' কমান্ড দিয়ে ভেঙে, নতুন গজানো '/opt/vim-6.2' ডিরেক্টরিতে গিয়ে '`./configure && make && make install`' করে

ইনস্টল করেছিলাম। (এই যে, ইনস্টল করার কায়দাটা অকারণেই একবার বলে নিলাম, এটা কিন্তু আপনাকে মনে পড়ানোর জন্যে যে গোটাটা আপনার মনে পড়ছে কিনা, কোথায় এই আলোচনাটা এসেছিল, মনে করুন তো।) এই গোটাটাই করেছিলাম রুট হয়ে, কারণ এটা ছিল আমার মেশিনের সুজে ৮.২ সিস্টেমে গ্লোবাল ইনস্টল। এই নতুন ভিমটা এবার আমার সিস্টেমের প্রত্যেক ব্যবহারকারীই পাবে, সেই অর্থে এটা আমার সিস্টেমের গ্লোবাল সেটিং। কিন্তু এই 'লোকাল-গ্লোবাল' বাগধারাটা আদতে কোথা থেকে এসেছিল সেটাও মাথায় রাখবেন, একটা নেটওয়ার্কিত সিস্টেমে একটা মেশিন হল লোকাল, গোটা নেটওয়ার্কটা গ্লোবাল। একজন সাধারণ ব্যবহারকারী শুধু লোকাল ইউজার প্রোগ্রামগুলোই চালাতে পারেন। লোকাল বা গ্লোবাল কোনোরকম সিস্টেম প্রোগ্রামই তার অধিকারের মধ্যে পড়েনা, এমনকি গ্লোবাল ইউজার প্রোগ্রামও না, সেগুলোর জন্যেও গোটা সিস্টেমব্যাপী কিছু বদল এবং কনফিগারেশনের দরকার পড়ে তা তার অধিকারের মধ্যে নেই। তার অধিকার তার হোম ডিরেক্টরিটুকু। এমনকি সেখানেও রুট যে কোনো বদল যে কোনো সময়েই ঘটাতে পারে।

ইউজার বা সিস্টেম দুরকম প্রোগ্রামই তাদের চালানো মাত্র নিজেদের কনফিগারেশন ফাইল একবার পড়ে নেয়। কোনো কোনো সিস্টেম প্রোগ্রাম চালু হয় একদম সিস্টেম বুট করার সময়েই, তাই তাদের কনফিগারেশন ফাইলও পড়া হয় সেই সময়েই। '/etc' ডিরেক্টরির সেইসব কনফিগারেশন ফাইল নিয়ে আমরা আলোচনা করেছি। ইউজার প্রোগ্রামগুলোরও একটা করে কনফিগারেশন ফাইল রাখা থাকে বা থাকতে পারে '/etc' ডিরেক্টরিতেই। প্রোগ্রামটা চালু হওয়ার সময় সেই ফাইল পড়ে নেবে। সেই ফাইল থেকে প্রোগ্রামটা পড়ে নেবে ডিফল্ট কনফিগারেশন। তারপরে, ব্যবহারকারী যদি নিজের কোনো বদল আনতে চায় সেটা ঘটাবে তার নিজের হোম ডিরেক্টরিতে রাখা বাড়তি একটা ব্যক্তিগত কনফিগারেশন ফাইলে। এই ফাইল মোতাবেক বদলটা দেখা দেবে শুধু ওই প্রোগ্রামের ওই ব্যবহারকারীর ব্যক্তিগত বা লোকাল ব্যবহারেই। প্রোগ্রামটার গ্লোবাল ব্যবহার নিয়ন্ত্রিত হবে '/etc' ডিরেক্টরির ওই গ্লোবাল কনফিগারেশন দিয়ে। আর যদি প্রোগ্রামটা গ্লোবালি ইনস্টল না-করে, শুধু একক ব্যবহারকারীর জন্যে লোকালি, তার হোম ডিরেক্টরিতেই ইনস্টল করা হয়ে থাকে তাহলে তো চুকেই গেল। কোনো কোনো প্রোগ্রামে তা হয় বৈকি। ফন্টের বেলাতেও হয়।

ধরুন ইম্যাক্স। এর কনফিগারেশনের জন্যে গ্লোবালি দেওয়া আছে '/etc/skel/.gnu-emacs' বা '/etc/skel/.emacs'। দেখুন, দুটোই ডটানন ফাইল, শুরুতে একটা করে বিন্দু বা ডট বা '.' আছে। আর ব্যক্তিগত ব্যবহারকারী 'dd'-র হোম ডিরেক্টরিতে বা '/home/dd'-তে আছে '.gnu-emacs' বা '.emacs'। যা মূলত '/etc/skel' ডিরেক্টরির ওই গ্লোবাল কনফিগারেশন ফাইলদুটোরই কপি। এবার পিউ ইম্যাক্সে সি-প্রোগ্রাম লেখা বা কম্পাইল করার জন্যে কিছু সুবিধেজনক বদল এনেছিল, যা লিখে নিয়েছিল ওর হোম ডিরেক্টরির '/home/piu/.emacs' ফাইলে। সেটা আমার হোম ডিরেক্টরির '/home/dd/.emacs' ফাইলে না-থাকায় সেই ব্যাপারগুলো আমি ইম্যাক্স ব্যবহার করার সময় ঘটেনা। এবার ভাবুন ইম্যাক্স তাহলে কী করছে — যেই কেউ 'emacs' মন্তর পড়ে ইম্যাক্স প্রোগ্রামটাকে জাগিয়ে তুলছে, ইম্যাক্স পড়ছে '~' ডিরেক্টরি থেকে ব্যক্তিগত কনফিগারেশন মানে '~/.emacs' ফাইল। এই '~' একটা চিহ্ন যা দিয়ে ব্যাশ শেল কোনো ব্যবহারকারীর হোম ডিরেক্টরিকে চিনে নেয়। তার মানে প্রোগ্রামটা যখন আমি মানে ইউজার 'dd' চালাচ্ছে তখন '~' হল '/home/dd'। আর যখন পিউ চালাচ্ছে, তখন '~' হল '/home/piu'। আর যদি কোনো ব্যক্তিগতভাবে বদলানো '~/.emacs' ফাইল না পায় তাহলে পড়ছে '/etc/skel' ডিরেক্টরি থেকে গ্লোবাল কনফিগারেশন। একটু বাদেই এইসব '~' জাতীয় খ্যাঁচগুলো ভালো করে বুঝতে হবে আমাদের।

আর একটা ধরুন আমার প্রিয় প্রোগ্রাম 'mplayer'। যাদুপর্বতের ওই আনুভূমিক আর উল্লম্বের রসিকতাটাকে একটু বদলে নিয়ে বলাই যায়, যখনই আমি অক্ষর দেখছি-না, মানে লিখছি বা পড়ছি না, তখনই আমি সিনেমা দেখছি। আর আমাদের তথাগত তো সোনার ছেলে, কী যে সব ভালো ভালো ফিল্ম আজকাল ওর কল্যাণে দেখতে পারছি আমি, সে আর বলার নয়। এবার এই 'mplayer' কাজ করে মূলত গ্লোবাল কনফিগারেশন ফাইল '/etc/mplayer.conf' দিয়ে, আর কোনো বাড়তি পছন্দ আপনি রেখে দিতে পারেন '~/.mplayer' ডিরেক্টরির '~/.mplayer/config' ফাইলে। এই '~/.mplayer/config' ফাইলটা আসলে '/etc/mplayer.conf' ফাইলেরই একটা কপি করে দিতে পারেন আপনি, তার পরে তাতে প্রয়োজনীয় পছন্দমত বদলগুলো ঘটালেন। দুটো ফাইলই যদি থাকে সিস্টেমে, এবং দুটোর সবকিছু না মেলে, তখন 'mplayer' মানবে '~/.mplayer/config' ফাইলে দেওয়া নিদানটাই। আর, শুধু এই দুটো জায়গাই না,

আর একটা তৃতীয় জায়গাও আছে কনফিগারেশন ফাইল রাখার, যে তিনটে জায়গাকেই খোঁজে 'mplayer', পরপর। এই তৃতীয় জায়গাটা হল '/usr/local/etc/mplayer.conf'। গ্নু-লিনাক্স ফাইলসিস্টেম হায়েরার্কি নিয়ে আমাদের আলোচনা থেকে দেখুন তো আপনি এই তৃতীয় জায়গাটার চক্করটা বুঝতে পারছেন কিনা। '/etc/mplayer.conf' ফাইলে পরপর দেওয়া আছে বিভিন্ন ধরনের সেটিং, শুধু তার শেষটা অনবদ্য, সেটা তুলে দেওয়ার লোভটা সামলাতে পারছি না। এই অংশটায় বলা আছে, শুধু উপরে দেওয়া ওই তিনটে জায়গাই নয়, যে কোনো জায়গায় যে কোনো ডিরেক্টরিতে আপনি কনফিগারেশন ফাইলটা রাখতে পারেন যদি চান, শুধু তার পুরো পথটা দিয়ে দিতে হবে, আর তখন এই ডিফল্ট ফাইলটা উড়িয়ে দিতে হবে, নয়ত 'mplayer' এই ডিফল্ট ফাইলটাই পড়বে। বলে একটু মিচকি হেসেছেও, মানে মিচকি হাসির টেক্সট প্রতিরূপটা ':)'। আর এরপরেই হল সেই অনবদ্য শেষ লাইন, যেখানে সেই পথটা লিখে দেখিয়ে দেওয়া আছে, আরটিএফএম শব্দটা মনে আছে তো?

```
## You can also include other configfiles
## Specify full path!
## Delete this default :)
#include = /home/gabucino/.mplayer/i_did_not_RTFM_carefully_enough...
```

ব্যবহারকারী বা ইউজারের স্তরে যে কোনো প্রোগ্রামই মোটামুটি প্রথমে খোঁজে '/etc' ডিরেক্টরির কনফিগারেশন ফাইল, সেখান থেকে পড়ে নেয় ডিফল্ট কনফিগারেশন। তারপরে ইউজার সেই প্রোগ্রামের কোনো কাস্টমাইজেশন বা নিজের পছন্দ অনুযায়ী বদল ঘটিয়েছে কিনা সেটা দেখে নেয় কোনো ডটানন ফাইলে বা আরসি ফাইলে দেওয়া কনফিগারেশন থেকে। এই আরসি ফাইল নিয়ে একটা মজা ঘটল। আমার আরসি বলতে মনে হচ্ছিল রিসোর্সের সঙ্গে কোনো একটা সম্পর্ক আছে, কিন্তু ঠিক মনে করতে পারছিলাম না, কোলকাতা লাগের মেইলিং লিস্টে, মানে, 'ilug-cal@ilug-cal.org' ইমেল আইডিতে, একটা চিঠি দিলাম, কেউ জানে কিনা, আরসি (rc) ঠিক কিসের অ্যাক্রোনিম। কিন্তু মেলটা পাঠানোর পরপরই 'www.faq.org' সাইটে এরিক রেমন্ডস-এর জার্গন ফাইলে পেয়ে গেলাম সংজ্ঞাটা। আমরা চার নম্বর দিনের ৭ নম্বর সেকশনে ১৯৬২-তে এমআইটির সিটিএসএস বা কম্পেটিবল টাইম শেয়ারিং সিস্টেমের কথা বলেছিলাম মনে আছে? তাতে সিস্টেম চালু হওয়ার সময়ে ব্যবহৃত রানকম (runcom) ফাইল থেকে এসেছে নামটা। এবং মেলিং লিস্টেই তথাগত লিখল, ও-ও আমারই মত, মনে করত, রিসোর্স কনফিগারেশন গোছের কোনো কথার অ্যাক্রোনিম এটা। এবং দেখুন, দুজনেই একই কথা ভাবছি মানে, যতদূর সম্ভব কোথাও একটা পেয়েছি নিশ্চয়ই। তা যাই হোক, সিটিএসএস থেকে আসা যে এই 'rc' নামটা এখন লাগু হয়ে গেছে যে কোনো বুটকালীন চলা স্ক্রিপ্টের জন্যেই, একটা কমন-নাউন হিশেবে।

সাধারন ব্যবহারকারী বা ইউজারের স্তরে এই ডট ফাইল এবং আরসি ফাইলের মূল প্রয়োজনীয়তার জায়গাটা কিন্তু রুট এবং ইউজারের অধিকারের পার্থক্য। প্রথমত, ন নম্বর দিনে আপনারা দেখেছেন, '/etc' ডিরেক্টরির বিভিন্ন কনফিগারেশন ফাইলের সঙ্গে বদলে ফেলার মত সহজ সম্পর্কে আসাটা শুধু শ্রমসাধ্যই নয়, যথেষ্ট গা-ছমছমেও বটে। বদলাব যে, কিছু যদি ঘেঁটে যায়? আর একটা সিস্টেমে প্রতিটা জিনিষই তো অন্য জিনিষগুলোর সঙ্গে অতিনির্ণয়ের ওভারডিটারমিনেশনের সম্পর্কে আছে, প্রতিটি কিছুই অন্য প্রতিটি কিছুর দ্বারা নির্ণীত এবং নির্মিত। তার মানে একটা কোনো বেমতলব বদল কতদূর অব্দি কোনটাকে ঘেঁটে দিতে পারে সেটা সম্পর্কে একটা আন্দাজ তৈরি হওয়ারও দরকার পড়ে। এবার ধরুন কোনো একটা প্রোগ্রামের একটা বিশেষ ব্যবহার আপনার পছন্দ হচ্ছেনা, আপনি সেটা বদলাতে চান, এবং জানেন যে সেটা বদলানো যায়, তখন খুব সহজেই যেটা করতে পারেন তা হল নিজের হোম ডিরেক্টরির কনফিগারেশন ফাইলটা বদলে নেওয়া, আর সেটা অনাবশ্যক ভাবে ঘেঁটে গেলে, আবার মূল '/etc' ডিরেক্টরির কনফিগারেশন ফাইলটা বদলানো ফাইলের জায়গায় পুনঃস্থাপিত করে দেওয়া। আরো একটা সমস্যা থাকে বহু মেশিনের বহু ইউজারের সিস্টেমে, সেটা এই যে, '/etc' ডিরেক্টরির ভিতরে কোনো ফাইলে হাত দেওয়ার অধিকার তো থাকবে না আপনার। ওটা গ্লোবাল সিস্টেম, তাই রুটের এলাকা। একা একলা একটা পিসিতে অবশ্য এই ঝামেলাটা থাকেনা। এই সব কারণেই এই দুই সেট ফাইল, একটা গ্লোবাল কনফিগারেশন, '/etc' ডিরেক্টরিতে, অন্যটা লোকাল, ইউজারের হোমে, ডট ফাইল বা আরসি ফাইল। এই ফাইলগুলোর বিষয়ে সামগ্রিকভাবে আর কোনো আলোচনায় যাচ্ছিনা, কারণ কিছু ফাইলের কথা তো বললাম, এরকম নামের মিল থেকে বুঝে নিতে পারবেন, আর অনেকগুলোই দখবেন এক্স-উইনডোজ সংক্রান্ত, মানে, আমাদের এই পাঠমালার এলাকার বাইরে। ও, কী ভাগ্যি যে

তখন বাদ দিয়েছিলাম। অবশ্য আমি খুব একটা পারতামও না বুঝিয়ে বলতে, নিজেই তো আন্দাজে মান্দাজে করি। বেশিরভাগই। দু-একটা কাজ অবশ্য খুব ইন্টারেস্টিং লাগে বেশ, যেমন এক্সউইনডোজের কনফিগারেশন ফাইল 'XF86Config'-টা বদলানো। ফাইলটা থাকে '/etc/X11' ডিরেক্টরিতে। ন নম্বর দিনে আমাদের ফাইলসিস্টেম হায়েরার্কির আলোচনা থেকে মনে করে দেখুন তো ঠিকানাটা আমাদের দেওয়া ছকের সঙ্গে মিলিয়ে নিতে পারছেন কিনা। যাকগে, কেরমে কেরমে আমাদের শেষের শুরু চলে এল, এবারে ব্যাশ শেলের কনফিগারেশন ফাইল। এই সেকশনে আসতেই পারত, সেই ডট ফাইল বা আরসি ফাইল, ইচ্ছে করেই পরের সেকশনে নিয়ে গেলাম। শুধু একটা কথা, আগের দিন ভুলে গেছিলাম কনফিগারেশন ফাইলের বৃত্তান্ত লেখার সময় যে লেখাগুলো থেকে সাহায্য পেয়েছিলাম তাদের রেফারেন্স দিতে। সেইটা দিয়ে রাখি, আর মূল রেফারেন্স তো দেখেছেনই নিজের মেশিনের সুজে সিস্টেমের ডিরেক্টরিগুলো। এই বাইরে থেকে সাহায্যগুলো সবই নেট থেকে পাওয়া, মালগুলো আমার কাছে নামানোই আছে, যদি কারুর পড়তে ইচ্ছে হয়, আমারই মত যার নেট করার সময় পয়সা গুণতে হয়, মেল করুন আমায়, কিছু একটা করা যাবে। এই বইটা লিখতে লিখতে দিন শূন্য থেকে শুরু করে নেট থেকে পাওয়া যত লেখা কাজে লাগিয়েছি, সবই রাখা আছে আমার '/mnt/arkive/lesson.bkp' ডিরেক্টরিতে।

http://www.comptechdoc.org/os/linux/howlinuxworks/index.html
Linux Startup Scripts by kimo@debian.lib.monash.edu.au
http://www.comptechdoc.org/os/linux/usersguide/index.html
ftp://www6.software.ibm.com/software/developer/library/l-config.pdf

২।। ব্যাশ শেলের লোকাল কনফিগারেশন ফাইল

ব্যাশ শেলের ব্যাশ নামের ব্যাখ্যা তো আগেই এসেছে — বর্ন-এগেন-শেল (**B**ourne-**A**gain-**SH**ell)। বেল ল্যাবরেটরির ইউনিক্স রিসার্চ ভার্শন সেভেনের জন্যে বর্ন শেল লিখেছিলেন স্টিভ বর্ন। তার পর থেকে আরো অনেক শেল লেখা হয়েছে। কর্ন-শেল, সি-শেল ইত্যাদি। মূল বর্ন শেল যা যা পারত তা সবই পারে ব্যাশ, এবং আরো বেশি কিছু পারে। বর্ন শেলের জন্যে লেখা প্রোগ্রাম ব্যাশেও চালানো যায়। গ্নু-লিনাক্সে ডিফল্ট শেল এই ব্যাশ। যা সিস্টেম এবং ব্যবহারকারীর মধ্যে মাধ্যমের কাজ করে। ইউজারের কাছ থেকে আদেশ পেয়ে জেগে ওঠে ব্যাশ, পাঁচ নম্বর দিনের থেকে মনে করুন, এবং সেই আদেশ পাঠিয়ে দেয় কারনেলের কাছে, তারপর কাজ সমাপ্ত হলে তার ফলাফল ইউজারের কাছে পৌঁছে দিয়ে ফের ঘুমোতে চলে যায়। এর পরের সেকশনে শেলকে অনেকটা বিশদে বুঝতে হবে আমাদের, এখন আমরা এই ব্যাশ শেলের কনফিগারেশন ফাইলগুলো বোঝার চেষ্টা করব।

ব্যাশ কনফিগার করা হয় মূলত পাঁচটা ফাইল দিয়ে। '/etc/profile', '/etc/bashrc', '~/.bash_profile', '~/.bashrc' এবং '~/.bash_logout'। এর মধ্যে প্রথম দুটো গ্লোবাল কনফিগারেশন ফাইল, এদের কথায় আমরা আসছি এর পরের সেকশনে। পরের তিনটে হল একজন একক ইউজারের লোকাল ফাইল, এর সবকটা ফাইলই যে সবসময় থাকবে তা নয়, কিন্তু চাইলে এদের যে কাউকেই ব্যবহার করা যায়। যেমন, আমার মেশিনের সুজে সিস্টেমে আমার হোম ডিরেক্টরিতে সমস্ত ডটানন ফাইলের তালিকার মধ্যে ব্যাশ সংক্রান্ত ফাইল আছে '~/.bash_history', '~/.bashrc' এবং '~/.profile'। এই শেষটা মানে '~/.profile' হল ওই '~/.bash_profile' ফাইলেরই নামান্তর। আর প্রথম ফাইলটা, '~/.bash_history', সাত নম্বর দিনের ২.৩ সেকশন থেকে আমাদের চেনা, এখানে আমাদের আদেশের ইতিহাস থাকে, এটার কথায় আমরা পরে আসছি। কোনো '~/.bash_logout' ফাইল সুজেতে থাকেই না, তবে চাইলে কোনো ইউজার বানিয়ে নিতেই পারে। সুজেতে আমার হোম ডিরেক্টরি মানে '/home/dd' থেকে '.bashrc' আর '.profile' — এই ডটানন ফাইলদুটো তুলে দেওয়া যাক। খেয়াল করুন, এই ডিরেক্টরিটা কিন্তু ইউজার 'dd'-র বেলায় '~'। প্রথমে '/home/dd/.bashrc' —

```
# There are 3 different types of shells in bash:
# The login shell, normal shell and interactive shell.
# Login shells read ~/.profile and interactive shells read ~/.bashrc;
test -s ~/.alias && . ~/.alias
if [ -x /usr/bin/fortune ] ; then
    echo -e '\n***\n'>>F;/usr/bin/fortune -os | tee -a F | cat
fi
```

এর শেষ চারটে লাইন আপাতত খ্যামা দিন, পরে আসছি, শুধু প্রথম চারটে লাইন, কমেন্ট লাইন, '#' লাগানো, আপাতত আপনার হজমনীয়। দেখুন, এইমাত্র যা বললাম, তার সঙ্গে মেলাতে পারছেন? এবার আসুন '/home/dd/.profile' পড়া যাক —

```
# This file is read each time a login shell is started.
# All other interactive shells will only read .bashrc;
test -z "$PROFILEREAD" && . /etc/profile
if [ -x /usr/bin/fortune ] ; then
    echo -e '\n***\n'>>F;/usr/bin/fortune -s | tee -a F | cat
fi
```

এখানেও আপাতত শুধু প্রথম দুটো লাইন পড়ুন। শুধু একটা মজা দেখুন, শেষ তিনটে লাইন দুটো ফাইলেই হুবহু এক — স্বাভাবিক, ওটা আমার অবদান, আর যা হয়, নিজের অবদান মাত্র একবার ঘোষণা করে সুখ কই? শুধু ফরচুনের অপশনে, '.bashrc' ফাইলে আছে '`/usr/bin/fortune -os`', আর '.profile' ফাইলে আছে '`-s`'। এটার মানে আর কিছু নয়, এইমাত্র কমেন্ট লাইনে যা পড়লেন, লগ-ইনের সময়ে '.profile' পড়ে ব্যাশ শুধু একটা ছোট ফরচুন শোনায়, '`s`' মানে শর্ট বা ছোট। আর আমি যখন কোনো জ্যান্ত শেলকে ডেকে আনি, শেলে কোনো কাজ করার জন্যে, যাকে ব্যাশের ম্যানপেজের ভাষায় বলে ইন্টারাক্টিভ ইনভোকিং, এক্স-উইনডোজে থাকাকালীন কোনো টার্মিনাল খোলা মানেও তাই, তখন ব্যাশ পড়ে নেয় '.bashrc', তখন শোনায় ওই আপত্তিকর বাণী, '`o`' মানে অফেনসিভ। একটু পরে ওই বাদ দেওয়া লাইনগুলোর মানে আর এই জায়গাগুলোও আরো স্পষ্ট হয়ে উঠবে। কিন্তু এর মানে এই নয় যে '~/.bashrc' বা '~/.profile' ফাইলে শুধু এই-ই থাকবে, এছাড়া আরো বহু কিছু থাকতে পারে এবং থাকেও, ইউজারের ইচ্ছা অনুযায়ী। কী কী ভাবে কী আনা যায় সেখানে সেগুলোর আলোচনায় আমরা আসছি। আমি এখানে এদুটোকে তুলে জাস্ট আপনাদের একটু অভ্যস্ত করলাম, আর এদের মধ্যে রাখা ওই কমেন্ট লাইনগুলো পড়াতেও চাইছিলাম। সুজে সিস্টেমে কোনো '~/.bash_logout' নেই, চাইলেই রাখা যেত। কিন্তু আমার মেশিনের অন্য সিস্টেমের ইউজার 'dd'-র হোম ডিরেক্টরি থেকে '.bash_logout' ফাইলটা তুলে দিই —

```
# ~/.bash_logout
clear
```

এখানে দেখুন, আর কিছুই নেই, কিন্তু চাইলেই রাখা যেত। এই '~/.bash_logout' হল একজন ইউজার লগ-আউট করে যাওয়ার পূর্ব-মুহূর্তে ঠিক কী কী কাজ সিস্টেমকে করে নিতে হবে, তার তালিকা। যেহেতু এটা এই ইউজারের লোকাল, তাই একজনের '~/.bash_logout' ফাইলে অন্য ইউজারের কিছু এসে যায়না।

'~/.bash_profile' বা '~/.profile' ফাইলে থাকে একজন একক ইউজারের নিজস্ব ব্যাশ কনফিগারেশন। এই ইউজার যখন ব্যাশ চালায় তখন তার এনভায়রনমেন্ট ভ্যারিয়েবলগুলোকে নিয়ন্ত্রণ করে এই '~/.bash_profile'। গ্লোবাল সেটিং-এ '/etc/profile'-এর যে ভূমিকা, লোকাল সেটিং-এ সেই একই ভূমিকা '~/.profile' ফাইলের। '/etc/profile' নিয়ন্ত্রণ করে সমস্ত ব্যবহারকারীর ব্যাশকে, আর '~/.profile' নিয়ন্ত্রণ করে একজন ব্যক্তি ব্যবহারকারীর ব্যাশকে। '~/.bashrc' ফাইলও তেমনি নিয়ন্ত্রণ করে একজন ব্যক্তি ব্যবহারকারীর নিজের পছন্দমত অ্যালিয়াসগুলোকে এবং নিজের পছন্দমত বিভিন্ন ফাংশনের ব্যবহারকে। এই অ্যালিয়াসের কথায় আমরা একটু বাদেই আসছি।

৩।। ব্যাশের গ্লোবাল কনফিগারেশন

আমরা যে পাঁচটা ফাইলকে ব্যাশের কনফিগারেশন বলে উল্লেখ করলাম, '/etc/profile', '/etc/bashrc', '~/.bash_profile' (বা '~/.profile'), '~/.bashrc', এবং '~/.bash_login', তার মধ্যে দেখুন, প্রথম দুটো, '/etc' ডিরেক্টরিতে — এরা হল গ্লোবাল কনফিগারেশন। মানে, এই সিস্টেমে যারা যারা ব্যাশ ব্যবহার করে তাদের সবারই ব্যাশকে নিয়ন্ত্রণ করবে এই ফাইলদুটো। যেমন আমরা আগেই বলেছি, গ্লোবাল কনফিগারেশন, তাই এদের সাধারণত এই ডিরেক্টরিতেই পাওয়া যাওয়ার কথা। পরের তিনটে হল লোকাল কনফিগারেশন, মানে একজন ব্যক্তি ইউজারের নিজের ব্যাশ-ব্যাবহারকে কনফিগার করা যায় এদের দিয়ে। নিজের হোম ডিরেক্টরিতেই গোপন বা হিড্ন ফাইল হিশেবেই থাকে এরা, কারণ এরা ডটানন, মানে মুখে ডট আছে এদের। স্বাভাবিক রকমে '`ls`' করলে যাদের দেখা যায়না, '`ls -a`' কমান্ড দিলে, মানে অল বা সব ফাইল দেখাতে বললে তখন এদের দেখা যায়। আর যেমন বললাম,

যদি এদের সব ফাইলকে আপনি আপনার হোম ডিরেক্টরিতে না পান, কুছ পরোয়া নেই, নিজের সিন্ধু দর্শনের বিন্দু ফাইল নিজেই বানান।

আর যে কোনো সময়েই আপনার করা বদলে ঘেঁটে যাওয়া ব্যাশ ঠিক করে নেওয়ার জন্যে আপনার সিস্টেমে '/etc/profile' তো আছেই — ব্যাশের গ্লোবাল কনফিগারেশন। সাত নম্বর দিনে যে এনভায়রনমেন্ট ভ্যারিয়েবল বা গেরস্থালির চলরাশির কথা উল্লেখ করেছিলাম, মনে করে দেখুন, '`set`' কমান্ড থেকে পাওয়া ফলাফলকে '`less`' করে পড়ে পড়ে — সেই চলরাশিগুলোকে নিয়ন্ত্রণ করে এই '/etc/profile'। ব্যাশ থেকে ব্যাশের উপর দাঁড়িয়ে যে প্রোগ্রামগুলো চালাতে হয়, ব্যাশস্ক্রিপ্টগুলো যেমন, সেগুলোর চলা এবং কাজ করাকেও নিয়ন্ত্রণ করে। মানে, যে কোনো ইউজার যখনি কোনো ব্যাশ প্রোগ্রাম বা স্ক্রিপ্ট চালায় তার ঘর-গেরস্থালির আকার-আকৃতি, মানে এনভায়রনমেন্ট ভ্যারিয়েবলদের মান কী হবে সেটা বলে দেয়। আপনাদের মধ্যে যারা উইনডোজ এমিগ্রি, মানে উইনডোজ জগত থেকে গ্নু-লিনাক্সে অভিবাসী, তারা মনে করতে পারবেন, 'c:\autoexec.bat' ফাইল দিয়ে যে কাজগুলো করতেন, প্রম্পটের আকার বদলানো, কোন কোন ডিরেক্টরি কাজ করার পথনির্দেশে থাকবে, ইত্যাদি, তাদের সবগুলো কাজই করা যায় এই '/etc/profile' ফাইল দিয়ে, এবং, স্বাভাবিক ভাবেই, আরো অনেক বেশি কিছু করা যায়। এরকমই আর একটা ফাইল '/etc/bashrc' — ব্যাশের আর একটা গ্লোবাল কনফিগারেশন ফাইল। কী কী ওরফ বা অ্যালিয়াস থাকবে আপনার, বা ব্যাশ চালু হওয়ার সময় কী কী ফাংশন বা কমান্ড বা স্ক্রিপ্ট গোড়াতেই চালিয়ে বা এক্সিকিউট করে নেবে, তার হদিশগুলো থাকে এই '/etc/bashrc' ফাইলে। আর অ্যালিয়াস মানে শর্টকাট, অল্প কথায় ব্যাশকে বড় বড় আদেশ দিতে পারা। একটু দাঁড়ান, পরেও লাগবে, তাই আমরা একটু অ্যালিয়াস নিয়ে দু-চার কথা বলে নেব এর পরের সেকশনে। তারপরে আবার ফিরে আসব ব্যাশের কথায়। অ্যালিয়াস ছাড়া এই গ্লোবাল কনফিগারেশন ফাইল '/etc/bashrc'-এ আর যা থাকে তা হল ব্যাশ চালু হওয়ার সময় যে যে ফাংশন চালিয়ে নেওয়া তাদের তালিকা। এই ফাংশনগুলো সবই একটা একটা ছোট ছোট স্ক্রিপ্ট, ব্যাশ সঠিক জায়গায় যাদের দরকারমাফিক চালিয়ে নেয়। যদি একটা সিস্টেমে '/etc/bashrc' ফাইল না থাকে, তাহলে এই সমস্ত জিনিষ দেওয়া থাকে '/etc/profile' ফাইলেই। কখনো কখনো ফাইলের নামটা একটু ভিন্ন হতে পারে, যেমন আমার সুজে সিস্টেমে ফাইলটার নাম '/etc/bash.bashrc'। সেখানে '/etc/profile' ফাইলের সাইজ ২৮৫ লাইন আর '/etc/bash.bashrc' ফাইলের সাইজ ২৫৩ লাইন।

৪।। অ্যালিয়াস বা ওরফ

এই অ্যালিয়াস বা ওরফ মানে ফাটা-রতন বা কাটা-গমা বললেই পুলিশ যেমন চিনে যায় এর পিছনে পিতৃদত্ত নামে কমল চন্দ্র সাহা বা পরেশ মাইতি, মানে এলাকার কোনো স্বনামধন্য SমাজSবীকে, এবং এদের কেউ কোনো নতুনতর সমাজসেবায় হাত দেওয়া মাত্রই নিট সেই মস্তানের কাছে তোলার রেট বাড়ায়। কারণ, ওই ওরফ বা অ্যালিয়াসটার পিছনে মূল ঘুঘুটাকে সবচেয়ে ভালোভাবে চেনে পুলিশ, তার পিছনে তার দাদা কাম পিতাকেও, তার আমদানির কারণেই চিনে রাখতে হয়। তেমনি, ঠিক আমদানি নয়, কমান্ড প্রম্পটে আপনার কাজকর্মের সুবিধের জন্যে সিস্টেমে আপনি কিছু ওরফ বা অ্যালিয়াস বানিয়ে নিতে পারেন। ডিস্ট্রোর তরফ থেকে নিজেরই বানানো ফাইল থাকে একটা। দাঁড়ান, উদাহরণ দিয়ে বোঝানো যাক। আপনার মেশিনে মোট কী কী অ্যালিয়াস ডিস্ট্রো থেকেই করা আছে সেটা জানতে হলে আপনাকে অ্যালিয়াসের ম্যানপেজ পড়তে হবে। কিন্তু অ্যালিয়াসের কোনো নিজের ম্যানপেজ নেই। অ্যালিয়াস হল ব্যাশের একটা আভ্যন্তরীন কাজ, বিল্ট-ইন ফাংশন। ব্যাশের ম্যানপেজ পাবেন ম্যানের সেকশন একে। আগের দিন আমরা যে ম্যানুয়ালের ওয়েবপেজগুলো বানিয়েছি তার থেকে 'bash.1.html'-টা খুলুন আপনার ব্রাউজার দিয়ে। অনেকটা পরের দিকে দেখুন একটা গোটা সেকশনই আছে 'SHELL BUILTIN COMMANDS' নামে। তার ভিতরেই পাবেন 'alias'-এর বিবরণ। দেখুন, দেওয়া আছে, কমান্ড প্রম্পটে শুধু '`alias`' বা '`alias -p`' কমান্ড আপনাকে আপনার সিস্টেমে গোটা অ্যালিয়াসের তালিকাটা দেখাবে। এবার আমার মেশিনের সুজে সিস্টেমের এই তালিকাটাকে রিডাইরেক্ট করে একটা ফাইলে এনে তার থেকে কয়েকটা লাইন তুলে দিই।

```
alias l='ls -alF'
alias la='ls -la'
alias ll='ls -l'
alias ls-l='ls -l'
```

এখানে শুধু 'ls' সংক্রান্ত কয়েকটা অ্যালিয়াস তুলে দিলাম, আছে কিন্তু প্রচুর। দেখুন, পড়ুন, মনে রাখুন। এবার এর মধ্যে আমাদের খুব পরিচিত হল দু-নম্বরটা। লিংক বুঝতে গিয়ে আমরা বারবার '`ls -l`' ব্যবহার করেছি। তার সঙ্গে আর একটা অপশন যোগ হয়েছে, '`a`'। সেটাও আমাদের চেনা, অল বা সমস্ত ফাইল দেখানোর, গোপন বা হিড্‌ন গুলোকেও। এবার দেখুন, অ্যালিয়াসটা তৈরি করা আছে বলে আমি কমান্ড প্রম্পটে '`la`' লিখে এন্টার মারলেই ও আসলে যে আদেশটা পালন করবে সেটা হল '`ls -la`'। কারণ, '/etc/bashrc' ফাইলে সেইরকমই বলে দেওয়া আছে।

এরকম আপনি নিজেও বানিয়ে নিতে পারেন। ধরুন আপনি চাইছেন, আপনি আপনার নতুন প্রবন্ধে হাত দেওয়ামাত্র ব্যাকগ্রাউন্ডে গান বাজতে শুরু করবে। গান না-শুনে লেখা যায়? আর গানগুলো আছে আপনার হোম ডিরেক্টরির ভিতর, 'music' বলে একটা ডিরেক্টরিতে, আর এগুলো সব 'mp3' ফাইল। 'mp3' কাকে বলে যদি না-জানেন তো বলে রাখি, সাধারণ অডিও মানে শব্দতথ্যের কোঁকড়ানো ছোটকরা একটা আকার। কতটা ছোট সেটার একটা আন্দাজ পাবেন নাম থেকে, এক একটার ভিতর তিনপিস করে ভুঁড়িদার হাতিমি সাইজের এমপি মানে মেম্বার অফ পার্লামেন্ট ধরে যায় — এমপি-র সংখ্যা তিন বা থ্রি। নামের এই বিশ্লেষণ যদি আপনার পছন্দ না-হয়, অন্য বিশ্লেষণটা কী সেটা নিজে খুঁজে নিন। আপনার সিস্টেমের হাউ-টু গুলোর ভিতরেই পেয়ে যাবেন।

এবং আপনি আপনার প্রবন্ধগুলো সব রেখেছেন আপনার হোম ডিরেক্টরির ভিতর 'probondho' ডিরেক্টরিতে। এবার আপনি এরকম নতুন একটা ওরফ বা অ্যালিয়াস বানাতে চান যা আপনার প্রবন্ধ লেখার ডিরেক্টরিতে নতুন প্রবন্ধ শুরু করার জন্যে আপনাকে ইম্যাক্স খুলে দেবে, এবং একই সঙ্গে ব্যাকগ্রাউন্ডে গান বাজাতে শুরু করবে, তাহলে কমান্ড দিন, '`alias likhi='mpg123 ~/music/*.mp3 & cd ~/probondho;emacs'`' এবং এন্টার মারুন। আর একবার '`alias`' কমান্ডটা মেরে দেখে নিন, এবার আগেরগুলোর সঙ্গে ও আপনার এই নতুন বানানো অ্যালিয়াসটাও যোগ করে দিয়েছে। এবার থেকে '`likhi`' কমান্ড দিলেই ব্যাশ আপনাকে এনে দেবে '~/probondho' ডিরেক্টরিতে এবং ইম্যাক্স খুলে দেবে আপনার জন্যে, আর ব্যাকগ্রাউন্ডে বাজতে থাকবে আপনার '~/music' ডিরেক্টরির '*.mp3' ফাইলগুলো একের পর এক। গানটা কিন্তু সত্যিই ব্যাকগ্রাউন্ডে চলে গেছে। আমাদের দেওয়া আদেশের মধ্যে '`&`' চিহ্নটা থাকায় ব্যাশ এটা করেছে। লিখতে লিখতে গানটা যদি কখনো বন্ধ করতে চান, তাহলে কিন্তু সরাসরি হবেনা, প্রথমে আপনাকে কাজ করতে থাকা ইম্যাক্সটাকে ব্যাকগ্রাউন্ডে নিয়ে যেতে হবে, '<Ctrl><Z>' সুইচ টিপে। দেখবেন ইম্যাক্সটা হাওয়া হয়ে গিয়ে কমান্ড প্রম্পট ফেরত এসেছে, কিন্তু ইম্যাক্স তাই বলে আদিউ জানিয়ে মহাপ্রস্থানে চলে গেছে তা কিন্তু নয়, জাস্ট ওয়াভোয়া বলেছে, এখুনি একবার '`fg`' লিখে এন্টার মারলেই ফেরত চলে আসবে স্ক্রিনে। ইম্যাক্সটা আসলে ব্যাকগ্রাউন্ডে চলে গেছে, বাজতে থাকা গানের ওই 'mpg123' প্রোগ্রামের মত। ইম্যাক্সকে যেভাবে একবার স্রেফ '`fg`' সেধেই ফেরত আনতে পারলাম আমরা, 'mpg123' প্রোগ্রামটা কিন্তু তা হবেনা। একেও ফোরগ্রাউন্ডে আনতে হবে '`fg`' মেরেই। কিন্তু, এখানে একটু মৃদু জাটিল্য আছে। এখন কিন্তু ব্যাকগ্রাউন্ডে একটা নয়, দুটো জব বা কাজকে আপনি পাঠিয়েছেন। তাহলে? '`fg`' মারলে কে উঠে আসবে ফোরগ্রাউন্ডে? জব এক না জব দুই? আসবে শেষ যাকে আপনি ব্যাকগ্রাউন্ডে পাঠিয়েছিলেন, মানে এই উদাহরণের ইম্যাক্স। তাহলে, গান থামাবেন কী করে? 'পায়েলিয়া গান থামা এবার' গাইবেন? বেগম আখতারের মত গাইতে পারলে সিস্টেম কখনো কখনো রিঅ্যাক্ট করে বলেই শুনেছি, কিন্তু আমরা যারা 'সুরের গুরু দাও গো সুরের দীক্ষা' গাইলেই অহল্যা জেগে ওঠে পাষাণ ফুঁড়ে, ট্রাফিক থেমে যায়, এবং নিকটতম অ্যামওয়েজ বিক্রেতা নাগা-সন্ন্যাসের সংকল্পে হিমালয়ের টিকিট কাটে, তাদের কী হবে?

তাদের জন্যে আছে জবস। কমান্ড দিন '`jobs`'। দেখুন, একটা তালিকা ফুটে উঠেছে। ফল্গুর বালির তলায় অন্তঃশীলা জলের প্রবাহের মত আপাতঅদৃশ্য কাজের মিছিল, হায় বাংলার অর্থনীতিতেও যদি একবার জবস মারা যেত। প্রত্যেকটা জবের নাম পাশে একটা করে সংখ্যা। যদি আপনি আর কোনো কাজকে ইতিমধ্যে, মানে শেষবার লগ-ইন করার পর থেকে আর ব্যাকগ্রাউন্ডে না-পাঠিয়ে থাকেন, তাহলে দেখুন, তালিকায় স্পষ্ট দেখাচ্ছে আপনার ওই 'likhi' কমান্ড থেকে তৈরি হওয়া নেপথ্যসঙ্গীত 'mpg123' আর আপনার নিজের হাতে অন্দরে পাঠিয়ে দেওয়া 'emacs', পর পর, তাদের নম্বর দুই আর এক। ইম্যাক্স এক কারণ শেষ সক্রিয় ছিল সে, এবং তার আগে সক্রিয় ছিল 'mpg123', তাই তার নম্বর দুই। জবসের এরকম একটা সম্ভাব্য বাণী এখানে একটু বুঝে নেওয়া যাক।

```
[1]-  Running         mpg123 ~/music/*.mp3 &  (wd: ~/probondho)
[2]+  Stopped         emacs
```

এবার আপনি 'fg 1' কমান্ড দিলে ফোরগ্রাউন্ডে আসবে ইম্যাক্স, এবং 'fg 2' কমান্ড দিলে ফোরগ্রাউন্ডে আসবে 'mpg123'। এদের মধ্যে ইম্যাক্সকে টাটা জানানোর উপায় তো জানেনই, '<Ctrl><X><Ctrl><C>'। আর 'mpg123'-কে টাটা জানানোর উপায় হল '<Ctrl><C>'। তবে যেহেতু এখানে একাধিক গান পরপর কিউতে বা সারিতে দেওয়া আছে '~/music/*.mp3' করে, তার মানে '~/music' ডিরেক্টরিতে মোট যত '*.mp3' ফাইল আছে তাদের চালানোর আদেশ, তাই একবার '<Ctrl><C>' মারলে ব্যাশ মনে করবে আপনি পরের গানটা শুনতে চাইছেন। না-থেমে, পরপর দুবার '<Ctrl><C>' মারলে ও বেরিয়ে যাবে 'mpg123' থেকে। আরো উপায় থাকতে পারে, আমার ভালো মনে পড়ছে না, 'man mpg123' বা 'mpg123 --help' করে দেখে নিন।

জবস কমান্ডের যে বাণীটা আমরা তুলে দিলাম, তার দু-একটা বিষয় এখনো আমাদের খেয়াল করার আছে। যেমন দেখুন উপরের লাইনে একদম ডানদিকে ব্রাকেটের ভিতর 'wd' — কোন শব্দবন্ধকে ছোট করে এটা করা হয়েছে বুঝতে পারছেন কি? 'pwd' কমান্ডটা মনে আছে? বারবার ব্যবহার করেছি আমরা, সেটাও যদি মনে না-থাকে ম্যানুয়াল থেকে দেখুন, তারপরেও মনে না-পড়লে, একটা টোটকা বলে দিই, খুব কাজে লেগে যায়, যে কোনো ভুলে যাওয়া কথা মনে পড়ানোয় — নিজের বাঁদিকের কানটা নিজে কামড়ে দেখুন, তবে কানটা খুলে আনবেন না, শুধু দাঁত বাড়িয়ে কামড়াবেন। তবে, কানে কোনো পুরোনো ইনফেকশন থাকলে কিন্তু করতে নেই। আর একদম বাঁদিকে, দুটো লাইনেই, চৌকো ব্রাকেটের ডান-পাশেই একটা '-' আর একটা '+' চিহ্ন। এই দুটোর মানে কী, সেটাও আমি বলে দেবনা, শুধু বলতে পারি যেরকম ফোরগ্রাউন্ড বা 'fg' কমান্ডের সঙ্গে একটা এক বা দুই দিয়ে আমরা একটা চলমান কিন্তু আপাতঅদৃশ্য জবকে পুরোভূমিতে বা ফোরগ্রাউন্ডে আনছিলাম, সেই রকম এই দুটো চিহ্ন দিয়েও করা যায়। এর গোটাটাই এবং আরো অনেককিছু পাবেন 'bash'-এর ম্যানুয়াল পেজে, দেখুন 'Job control' নামে একটা আলাদা গোটা সেকশনই আছে ব্যাশের ম্যানুয়ালে। অ্যালিয়াস (alias) বা ওরফ যেমন ব্যাশ শেলের একটা অন্তর্বর্তী আদেশ বা শেল বিল্ট-ইন কমান্ড, সেরকম জবস (jobs) বা ক্রিয়াশীল কাজের তালিকা দেখানোটাও ব্যাশের ভিতরকার কাজগুলোর একটা।

এইমাত্র যে অ্যালিয়াসটা আপনি বানালেন সসঙ্গীত সাহিত্যচর্চার, সেটা কিন্তু অস্থায়ী অ্যালিয়াস। পুলিশের রেকর্ডে এখনো তার নাম তোলা হয়নি। পরেরবার লগ-ইন করে আপনি আর পাবেন না অ্যালিয়াসটা। তাহলে কী করবেন, পরের বার পেতে হলে? কোনো একটা কনফিগারেশন ফাইলে যোগ করে দেবেন। গ্লোবাল নয়, লোকাল ফাইল। আপনার শিল্পমনন অন্যদের উপর চাপানোর কোনো অধিকার আপনার নেই। কিন্তু নিজের জন্যে এটাকে তুলে রাখতেই পারেন, আপনার হোম ডিরেক্টরির ডটানন ফাইল '~/.bashrc' ইম্যাক্স দিয়ে খুলে তার একদম শেষে হুবহু একটু আগের আমাদের অ্যালিয়াস বানানোর গোটা লাইনটা টাইপ করে তুলে দিন। এমনকি যদি ফাইলটা এমনিতে আপনার হোমে নাও থাকে, একটু আলতো করে ছুঁয়ে দিন, টাচ কমান্ডটা তো মনে আছে, 'touch ~/.bashrc'। তারপর ইম্যাক্স দিয়ে খুলে টাইপ করে দিন। বা, সরাসরি 'cat>>~/.bashrc' কমান্ড দিয়ে কমান্ড প্রম্পটে গোটা লাইনটা টাইপ করে দিয়ে, '<Ctrl><D>' মেরে ক্যাট করা শেষ করুন। হিল্লে হয়ে গেল। এই '>>' অপারেটরটা মনে পড়ছে তো, শির জিনিষ, যদি ফাইল থাকে তো তার শেষে লাগিয়ে দেবে, আর না-থাকে তো বানিয়ে লাগিয়ে দেবে। এতে সুবিধেটা এই যে ওই নামে ফাইলটা থাকলেও সেটাকে উড়িয়ে দেবেনা, এমনিতে ক্যাট যা করে। এর পর থেকে প্রত্যেকবার লগ-ইন করার পর এই অ্যালিয়াসটা আপনা থেকেই পাবেন। তবে, '~/.bashrc' ফাইল এমনিতেই আপনার হোমে থাকলে, ডিফল্ট ব্যবস্থায় সেটাই থাকার কথা, একটা নিরাপত্তা ব্যবস্থা আগেই করে রাখুন, কমান্ড দিন, 'cp ~/.bashrc ~/.bashrc.bkp'। এতে সুবিধেটা এই যে, নিজে করতে গিয়ে ঘেঁটে ফেললে, পরে ফেরত পেয়ে যাবেন আদত '.bashrc' ফাইলটা '.bashrc.bkp' নামে। তখন পুরোনোটা ফেরত পেতে গেলে জাস্ট একটা কমান্ড দিন, 'mv ~/.bashrc.bkp .bashrc', মানে পুর্নব্যাশো-ভব করে দিলেন। কনফিগারেশন ফাইল নিয়ে নখরাবাজি করার আগে এই কবচকুণ্ডলটা সবসময় মনে রাখবেন। কারন, শিখতে গেলে সিস্টেম ঘাঁটবেই, সেই প্রাচীন প্রবাদটা তো শুনেছেন নিশ্চয়ই — "কঁপিতে শেখেনা কেহ না-ঘেঁটে মেশিন" — 'কঁপিতে' শব্দটা হল 'কম্পিউটার-করা' নামক ক্রিয়াপদের প্রাচীন মৈথিলী ব্রজবুলি আকার, সুনীতি চাটুজ্জের ওডিবিএল-এ দেওয়া আছে।

৫।। ব্যাশ এবং তার ভ্যারিয়েবল

একটু আগে ব্যাশের কনফিগারেশন ফাইলগুলো আলোচনা করতে গিয়ে বারবার এই এনভায়রনমেন্ট ভ্যারিয়েবলের কথা আসছিল। সাত নম্বর দিনে এই এনভায়রনমেন্ট ভ্যারিয়েবল বা প্রতিবেশ-চল কাকে বলে তা আমরা একটু ছুঁয়ে চলে এসেছিলাম, আজ আমরা দেখার চেষ্টা করব, কোথা সে পথের শেষ, ওহে চঞ্চল। চঞ্চল, মানে চলে বেড়ায়, মানে ভ্যারি করে বা ভ্যারিয়েবল, সেই জন্যেই এদের বলে চল বা চলরাশি। রাশিটা আসে গণিতে ব্যবহারের সূত্রে। শূন্য এক আর দুই নম্বর দিনে আমরা বারবার স্মৃতিভূমির কথা এনেছি, নানা ধরনের নানা আকারের স্মৃতিভূমি। কখনো তা হার্ডডিস্কের মত স্মৃতির ভাঁড়ার, কখনো র‍্যামের মত উদ্বায়ী স্মৃতি। পরে আট নম্বর দিনে এই র‍্যামকে দিয়ে গোটা কাজ করতে গিয়ে অপারেটিং সিস্টেমের কারনেলের কাছে যাতে স্মৃতি নামক রসদের যাতে টান না-পড়ে, সেই জন্যে, ১২৮ বা ২৫৬ বা ৫১২ যত এমবি ভৌত র‍্যামই থাক, তার সঙ্গে সঙ্গে পোঁ-ধরার মত একটা ভৌতিক স্মৃতি বা ভারচুয়াল মেমরির কথা বলেছি। সোয়াপ ফাইলের ফাইল সিস্টেমের কথা বলতে গিয়ে এবং হার্ডডিস্কের বিভিন্ন ধরনের ব্লকের আলোচনা করতে গিয়ে আমরা বলেছি, কী ভাবে একটা গ্নু-লিনাক্স সিস্টেম তার প্রয়োজনীয় সমস্ত কিছুকে এই ভারচুয়াল মেমরিতে তুলে নেয়, এবং লগ-আউট বা শাট-ডাউনের সময় সেই গোটা ভৌতিক ক্রিয়াকলাপটাকে ফ্লাশ করে, মানে ভৌত ডিস্কে লিখে দিয়ে বেরিয়ে আসে।

আমরা যখন পাঁচ নম্বর দিনে, 'dd' নামের ইউজার হয়ে '`echo $PATH`' মেরে একক ব্যক্তি ইউজার কোনো কমান্ড দিলে ব্যাশ শেল কোথায় কোথায় ওই কমান্ডে দেওয়া নামের প্রোগ্রাম ফাইলটা খুঁজবে তার তালিকা, মানে পথনির্দেশ দেখেছিলাম, বা, 'root' হয়ে মানে সুপার-ইউজার হয়ে ওই একই কমান্ড দিয়ে রুট ইউজারের শেলের পথনির্দেশ দেখেছিলাম, তখন ব্যাশ আসলে কী করছিল? যে ইউজারের হয়ে আমরা কমান্ডটা চালাচ্ছি, তার জন্যে, সিস্টেমের স্মৃতিভূমিতে সে 'PATH' নামের এক ফালি জমি তুলে রেখেছে। সেই জমিতে যা ফলে আছে সেই ফলনটা ইকো বা প্রতিধ্বনি করে দিচ্ছিল স্ক্রিনে। এই জমিটা হল 'PATH' নামের একটা ভ্যারিয়েবল। আর ফলনটা হল তার মান বা ভ্যালু, আমরা তো ইতিমধ্যেই জানি '$' দিয়ে ভ্যালুকে বোঝানো হয়। তাই '$PATH' হল 'PATH' নামক ভ্যারিয়েবলের ভ্যালু। এই দুটো পথনির্দেশ যে আলাদা হয় তাও আমরা জানি। দেখেছি বারবার। এবং এখন এটাও আমরা জানি রুটের খেতের তরমুজ কী করে এলেবেলে ইউজারের খেতে করমচা হয়ে যাচ্ছে। যে কোনো ইউজার যেই লগ-ইন করছে, সে এমনি ইউজার হোক, বা রুট হোক, বা এমনি ইউজার '`su`' করে রুট হোক, তার প্রত্যেকবারই একটা করে আলাদা আলাদা ব্যাশ শেল চালু হচ্ছে। দুই নম্বর দিনের বিভিন্ন ধরনের মাল্টিপ্লেক্সিং-এর আলোচনা মনে করুন, একই সঙ্গে চলমান রাশি রাশি প্রক্রিয়া বা প্রসেসের সঙ্গে একটা আদিম চিড়িয়াখানার সেই তুলনা। সেই প্রসঙ্গে বারবার এসেছে প্রসেসের বা প্রক্রিয়ার কথা — একটা প্রোগ্রাম ফাইলকে গ্নু-লিনাক্সে সরাসরি চালানো হয়না, তার একটা প্রতিরূপকে তুলে নেওয়া হয় ভৌতিক স্মৃতিতে। সেটাই একটা প্রসেস। ঠিক সেরকম, যখনই কোনো ইউজার লগ-ইন করে, বা কোনো ইউজার ইন্টারাক্টিভলি মানে জ্যান্ত পারস্পরিক রকমে ব্যাশকে চালায়, তখনই সিস্টেম ব্যাশ নামক প্রোগ্রাম ফাইলের একটা প্রতিরূপ নিজের স্মৃতিতে তুলে নেয়, জন্ম নেয় আর একটা প্রসেস বা প্রক্রিয়া। খুব সুন্দর করে এই প্রসেসের ব্যাপারটা বোঝানো আছে ট্যানেনবমের 'মডার্ন অপারেটিং সিস্টেমস' বইটাতে, যার রেফারেন্স আগেও একাধিকবার দিয়েছি আমরা। আর একান্তভাবে গ্নু-লিনাক্সের বেলায় এই ব্যাপারটা প্রাথমিকভাবে বোঝার জন্যে আছে এরিক রেমন্ডস-এর 'ইউনিক্স অ্যান্ড ইন্টারনেট ফান্ডামেন্টালস হাউ-টু', হাউ-টু গুলোর মধ্যেই পাবেন। প্রয়োজনীয় সমস্ত খুঁটিনাটিই সিস্টেমের ভিতরেই আছে, নিজে খুঁজে নিন।

যখনই একটা নতুন ব্যাশ-প্রক্রিয়া চালু হচ্ছে, সে এমনি ইউজারের জন্যে হোক, বা রুট ইউজারের জন্যেই হোক, ব্যাশ চালু হওয়ার আগে ওই কনফিগারেশন ফাইলগুলো পড়ে নিচ্ছে, এবং সেইমত প্রতিটা এনভায়রনমেন্ট ভ্যারিয়েবলের জন্যে একটা করে স্মৃতিখণ্ড বরাদ্দ করছে, আর যখনই কোনো প্রয়োজন পড়ছে, যেমন আমরা যখন পথনির্দেশ জানতে চাইলাম, ও সেখান থেকে পড়ে নিচ্ছে সেই প্রতিবেশ-চলর মান এবং তাকে ফুটিয়ে তুলছে স্ক্রিনে। ব্যাশে কিন্তু শুধু প্রতিবেশ-চল না, আরো এক রকমের চল বা ভ্যারিয়েবল হয়, তাকে বলে লোকাল বা আঞ্চলিক ভ্যারিয়েবল। এই আঞ্চলিক বা লোকাল ভ্যারিয়েবলগুলোকে বানিয়ে তোলা হয় একজন ব্যক্তি ইউজারের ইচ্ছা অনুযায়ী। বানিয়ে তোলাটা ঘটতে পারে দুই ভাবে। এক, ব্যক্তি ইউজার সরাসরি বানাল, কমান্ড প্রম্পটে কমান্ড দিয়ে। দুই, ব্যাশ চালু

হওয়ার সময় যে লোকাল কনফিগারেশন ফাইল পড়ে নেয়, ব্যক্তি ইউজারের হোম ডিরেক্টরি থেকে, সেই ফাইলগুলো মারফত।

সরাসরি আপনি যখনি ইন্টারাকটিভলি মানে 'bash' কমান্ড দিয়ে কোনো ব্যাশ চালু করলেন, এবং তার মধ্যে কোনো একটা ভ্যারিয়েবল বানিয়ে গুঁজে দিলেন, এই ব্যাশটা, মানে এই ব্যাশের এই চলমান প্রতিরূপ প্রক্রিয়ায় সেই ভ্যারিয়েবলের নামে একটা জমি তৈরি হল, এবং সেখানে তার মানটা গুঁজে দেওয়া হল। কিন্তু এই 'exit' কমান্ড দিয়ে বা '<Ctrl><D>' করে ব্যাশ থেকে বেরিয়ে যাওয়া মানে এই ভ্যারিয়েবলটারও ইন্তেকাল হয়ে গেল। এবার, একটা জিনিষ খেয়াল করুন। বেরিয়ে তো গেলেন এই ব্যাশ থেকে, কিন্তু বেরিয়ে গেলেন কোন চুলোয়? কেন? ব্যাশের বাইরে? না, সরি, সেটা যাওয়ার উপায় নেই, পাঁচ নম্বর দিনের আলোচনাটা মনে করুন, কোনো ইউজার লগ-ইন করে ঢোকা মানেই একপিস শেলের আধারে। শেলই সেই আবহ বা অ্যাম্বিয়েন্স, যার বাইরে আপনি বেরোতে পারেন না, বেরোতে পারেন একমাত্র সিস্টেম থেকে বেরিয়ে গেলেই, মানে, লগ-আউট করলেই। কোনো ইউজার লগ-ইন করার মুহূর্তেই চালু হয়ে গেছে প্রথমতম ব্যাশ-প্রক্রিয়াটা। তার পর থেকে যত নতুন নতুন ব্যাশ প্রক্রিয়া চালু করছে ওই ইউজার বা সিস্টেম, তার প্রত্যেকটাই ওই প্রথমতম ব্যাশ প্রক্রিয়ার ছানাপোনা।

এই জনিতৃ প্রক্রিয়া বা পেরেন্ট প্রসেস এবং সন্তান প্রক্রিয়া বা চাইল্ড প্রসেসের ধারণাটা গ্নু-লিনাক্স তথা যে কোনো ইউনিক্স সিস্টেমেই অত্যন্ত গুরুত্বপূর্ণ, এবং অতীব মজার। এবং এর জন্যেও আমার রেফারেন্স ট্যানেনবম। বইটা সত্যিই বেধড়ক। আমার মত যারা অটেকনিকাল রকমে, নিজে নিজে, হাফবয়েল ডিমের মত করে, কম্পিউটার শেখে তাদের জন্যে এর চেয়ে ভালো কোনো বই আমার চোখে পড়েনি। তবে কম্পিউটারের কটা বই-ই বা দেখেছি আমি? লিনাসের সঙ্গে যতই কুচুটেপনা করে থাকুন, মাস্টারমশাই হিশেবে ট্যানেনবম যে পুরো সাবলাইম তাতে কোনো সন্দেহ করে কোন শালা। ট্যানেনবমের স্তরের টেক্সটবই আমি আমার জীবনে আর দুটো মাত্র দেখেছি — স্যামুয়েলসন আর নর্ডহউসের ইকনমিক্স এবং কুরান্টের ক্যালকুলাস। সাসপেন্স থ্রিলারের মত উত্তেজক, টেক্সটবই লিখনা হ্যায় তো অ্যায়সা। একজন পাঠক তথা ছাত্রের গোটা চিন্তাপ্রক্রিয়াটাকেই এতোলবেতোল করে দেওয়া। আমার এই বইটায় বহু জায়গায় বহু বিষয় এসেছে যা সরাসরি ট্যানেনবম থেকে ঝাড়া। মানে যতটা ভালো আমি ঝাড়তে পেরেছি। তবে বইটাকে পড়া দরকার গ্নু-লিনাক্সে নিজের সিস্টেমের সঙ্গে মিলিয়ে মিলিয়ে। ওখানে তো ইউনিক্স উইনডোজ ওএসটু সোলারিস সবই আছে। আর নৈর্ব্যক্তিক একটা রকমে, প্রতিমুহূর্তে সব ধরনের অপারেটিং সিস্টেমের উদাহরণ টেনে টেনে।

যা বলছিলাম, আঞ্চলিক চল বা লোকাল ভ্যারিয়েবল, যাদের সরাসরি, ইন্টারাক্টিভলি, একজন ইউজার বানিয়ে তুলছেন। একটু আগে অ্যালিয়াস দিয়ে যে কাজটা করছিলাম আমরা, এবারে ব্যাশের লোকাল ভ্যারিয়েবল দিয়ে ওই একই কাজ করা যাক। 'likhi' নামের অ্যালিয়াসটা তৈরি করতে গিয়ে আপনি সমান চিহ্নের ডানদিকে যে অংশটা রেখেছিলেন, সেটাকেই এবার একটা ভ্যারিয়েবল বানিয়ে তার মান বা ভ্যালু হিশেবে দিয়ে দিতে হবে। কমান্ড দিন, 'lekho='mpg123 ~/music/*.mp3 & cd ~/probondho;emacs''। দুপাশে দুটো কোট চিহ্ন দিতে হল কারণ মধ্যে স্পেস-চিহ্ন এসেছে। যদি আমরা কোট বা উর্দ্ধ-কমা না-দিতাম, তো, ব্যাশ ভ্যারিয়েবলের মান হিশেবে নিয়ে নিত প্রথম স্পেস চিহ্নের আগে অব্দি। এবার এই যে লোকাল ভ্যারিয়েবল 'lekho' আমরা তৈরি করলাম, এর মান কত বলুন তো? এবং সেই মানটাকে ব্যাশ কী নামে চেনে? আপনার বলতে পারার কথা। না-পারলে চিন্তার কথা, কারণ, তাহলে আরো বহু জরুরি জিনিষই আপনার মনে নেই, আর একবার গোড়া থেকে পড়তে পড়তে আসুন। আমরা এখন 'echo $lekho' কমান্ড দিলেই, ব্যাশ স্ক্রিনে ফুটিয়ে তুলবে সমান চিহ্নের ডানদিকে দুটো কোট চিহ্নের মধ্যে গোটাটা, অবশ্যই কোট ছাড়া। কারণ '$lekho' হল 'lekho' নামের লোকাল ভ্যারিয়েবলের মানের ব্যাশীকৃত নাম। মূল মজাটা অবশ্য অন্যত্র। এবার কমান্ড প্রম্পটে নিছক '$lekho' টাইপ করে এন্টার মারুন। দেখুন, একটু আগে আপনি অ্যালিয়াস দিয়ে যে কাজটা করছিলেন, সেই গোটাটাই এখন এইভাবে করা যাচ্ছে। মানে, আমরা 'lekho' নামের লোকাল ভ্যারিয়েবল দিয়ে সেই কাজটাই করলাম, যেটা একটু আগে 'likhi' নামের অ্যালিয়াস দিয়ে করেছিলাম।

এবার দেখুন, এই সদ্য বানানো লোকাল ভ্যারিয়েবল কিন্তু একদমই এই বিশেষ ব্যাশ প্রক্রিয়ার লোকাল, একে লং-ডিসট্যান্স করবার জন্যে, দূরপাল্লায় ছোটানোর জন্যে, অন্য অন্য ব্যাশ প্রক্রিয়া যারা এর পরে সিস্টেমে শুরু হবে

তাদের কাছে পৌঁছে দেওয়ার জন্যে আপনাকে একে প্রদেশের বাইরে আন্তর্জাতিক স্তরে রপ্তানি করতে হবে। এটা যে একান্তই এই ব্যাশ প্রক্রিয়ার নিজস্ব, সেটা বোঝার উপায় হল, নিছক একবার 'bash' টাইপ করে এন্টার মারা। আপনার কমান্ড প্রম্পটে কোনো পার্থক্যই ঘটল না, কিন্তু সিস্টেমের কাছে মোট প্রক্রিয়ার মানচিত্রে আপনার অবস্থানটা বদলে গেল। এতক্ষণ আপনি ব্যাশেই ছিলেন, এখন গেলেন ব্যাশের-মধ্যে-ব্যাশে। এটাও ব্যাশ, কিন্তু একটা নতুন প্রক্রিয়া বা প্রসেস। এখানে আপনি '$lekho' কমান্ড দিন। কী পেলেন? ব্যাশের অজ্ঞতা — কমান্ড নট ফাউন্ড — ব্যাশ এই কমান্ডটা জানেই না। বেচারা জানবেই বা কী করে? আপনি তো রপ্তানি করেননি। এই ব্যাশের-মধ্যে-ব্যাশ থেকে বেরিয়ে যান। তার জন্যে 'exit' বা '<Ctrl><D>' তো আছেই। এবার আবার একবার '$lekho' কমান্ড দিয়ে দেখুন, ফের নিখুঁত চিনে যাচ্ছে কমান্ডটা, এমনকি আপনাকে গোটাটা দিতেও হবেনা, ছ-নম্বর দিনের সেই ব্যাশ কমপ্লিশন বা একটু টাইপ করার পর ট্যাব মারলে অন্যটুকু ব্যাশের নিজেই পুরন করে দেওয়ার ব্যাপারটা মনে আছে তো? আপনি কমান্ড প্রম্পটে '$le' টাইপ করে ট্যাব মারুন, হয় দেখবেন গোটা '$lekho'-টাই ও দেখিয়ে দিচ্ছে, নয়তো নিচে একাধিক সম্ভাবনা দেখাচ্ছে, যার মধ্যে একটা হল '$lekho'। মানে, এই ব্যাশ আমাদের বানানো লোকাল ভ্যারিয়েবলটাকে চেনে। সে তো চিনবেই, এর মধ্যে এমন কিছুই ক্যালি নেই, আমরা তো ভ্যারিয়েবলটা বানিয়েছিলাম এই ব্যাশেই। এবার একে এক্সপোর্ট করুন। কমান্ড দিন 'export lekho', এন্টার মারুন। ব্যাস। আপনি রপ্তানি করে দিলেন। এবার আগের মতই 'bash' টাইপ করে আর একটা ব্যাশ খুলুন, সেখানে দেখুন, এই '$lekho' কমান্ডটা দিব্য রয়েছে, মানে এক্সপোর্ট হয়ে গেছে এখন থেকে ব্যাশীতব্য যে কোনো ব্যাশের কাছেই।

কিন্তু এই রপ্তানি আপনি সরকারি ভাবে করেননি। রপ্তানি দপ্তরের কোনো বড়কর্তার বা তার জ্যেষ্ঠবাবা কোনো বড়মন্ত্রীর ঘুষ খাওয়ার কোনো সুযোগ আপনি এখনো করেননি, এদিকে রেগুলার দিনে রাতে রপ্তানি করে যাবেন, একি মামদোবাজি নাকি। সরকারি দপ্তরের ফাইল ঘুরিয়ে আনতে হবে এই রপ্তানিকে, যদি যখন তখন লগ-ইন করা মাত্র যে কোনো ব্যাশে আপনি এই ভ্যারিয়েবলটাকে খুঁজে পেতে চান। কিন্তু কোন ফাইল? অবশ্যই ব্যক্তি ইউজার হিশেবে আপনার লোকাল কনফিগারেশন ফাইল। যে লাইনটা দিয়ে আমরা 'lekho' ভ্যারিয়েবলটা বানিয়েছিলাম, সেই গোটা লাইনটা, কোট চিহ্ন সহ, নিট যোগ করে দিন আপনার হোমের '~/.bashrc' ফাইলে। এবার ঠিক অ্যালিয়াসের মতই, আপনার লগ-ইন করা থেকে শুরু করে যে কোনো ব্যাশেই পেয়ে যাবেন এই লোকাল ভ্যারিয়েবলটা, এবং তার মান আকারে এই '$lekho' কমান্ডটা। এর আগে অব্দি আমাদের বানানো লোকাল ভ্যারিয়েবল 'lekho' ছিল একটা অস্থায়ী ভ্যারিয়েবল এবং একটিমাত্র বিশেষ ব্যাশ-প্রক্রিয়ার নিজস্ব। একটা ব্যাশেই যার জন্ম ও মৃত্যু। এখন এই লোকাল ভ্যারিয়েবলটাকে স্থায়ী করে দেওয়া হল, '~/.bashrc' ফাইলে তার জন্মপ্রক্রিয়াটাকে ভরে দিয়ে। এবার আপনি যদি সিস্টেমের সুপারইউজারও হন, তাহলে এটাকে আপনি সিস্টেম জুড়ে যে কোনো ব্যবহারকারীর কাছেই প্রাপ্তব্য করে তুলতে পারেন। তখন আপনি আর একজন ইউজার হয়ে নিজের হোম ডিরেক্টরির '~/.bashrc' ফাইলে লাইনটা যোগ করবেন না। যোগ করবেন ব্যাশের গ্লোবাল কনফিগারেশন ফাইলে। সেরকম কোনো ফাইলের কথা আপনি জানেন নাকি? গ্লোবাল কনফিগারেশন ফাইলে যোগ করে দিলে তখন সেটা হয়ে যাবে গ্লোবাল ভ্যারিয়েবল, সিস্টেম জুড়ে যে কোনো ইউজারের বেলাতেই সত্যি। এবং স্থায়ী তো বটেই। যে কোনো ইউজারের যে কোনো ব্যাশ চালু হওয়ার আগে '/etc' ডিরেক্টরির কনফিগারেশন ফাইল পড়েই তাদের বানিয়ে নেবে। তখন সেটা মোট এনভায়রনমেন্টের বা প্রতিবেশেরই একটা অংশ হয়ে যাবে, তাকে আমরা ডাকব প্রতিবেশ চল বা এনভায়রনমেন্ট ভ্যারিয়েবল বলে।

ব্যাশের কাজ করার এই দুই ধরনের ভ্যারিয়েবল — এনভায়রনমেন্ট ভ্যারিয়েবল এবং লোকাল ভ্যারিয়েবল এরা ব্যাশের কাছে আসে দুই ভাবে। এনভায়রনমেন্ট ভ্যারিয়েবল বা প্রতিবেশ-চলদের গজিয়ে তোলে সিস্টেম, '/etc/profiles' বা '/etc/bashrc' গোছের গ্লোবাল কনফিগারেশন ফাইল অনুযায়ী। এদের মধ্যে পড়ে 'SHELL', 'PS1', 'PATH' ইত্যাদি। পরে আমরা বিশদ ভাবে আসছি এদের কথায়। আর লোকাল ভ্যারিয়েবল বা স্থানীয়-চলদের বানিয়ে তোলে ইউজার, একটা জ্যান্ত পারস্পরিক ইন্টারাক্টিভ ব্যাশ প্রক্রিয়ায়, নিজে কমান্ড প্রম্পটে কমান্ড দিয়ে, যেমন আমরা 'lekho'-কে বানালাম। বা একজন ইউজারের নিজের হোম ডিরেক্টরিতে থাকা লোকাল কনফিগারেশন ফাইল পড়ে সেই ইউজারের জন্যে চালু হওয়া ব্যাশ-প্রক্রিয়া এদের বানিয়ে তোলে। এই লোকাল ভ্যারিয়েবলদের হালহদিশ ভরা থাকে ইউজারের নিজের '~/.profile' বা '~/.bashrc' গোছের ফাইলে। ব্যাশ চালু হওয়ার সময় এই

ব্যক্তি ইউজারের লোকাল কনফিগারেশন অনুযায়ী এদের বানায়, যে কনফিগারেশনগুলো নিজের মর্জি বা প্রয়োজন মত বদলে নেয় একজন ব্যক্তি ব্যবহারকারী। ঠিক যে ভাবে পরে আমরা 'lekho' বানানোর আদেশকে স্থায়ী করার জন্যে কনফিগারেশন ফাইলে ভরে দিয়েছিলাম।

৬।। ব্যাশ ভ্যারিয়েবলের ঠিকুজি

একটু আগে কমান্ড প্রম্পটে যে ধরনের কমান্ড দিয়ে আমরা 'lekho' ভ্যারিয়েবলটা বানালাম, একে বলে ভ্যারিয়েবলের ডেফিনিশন বা সংজ্ঞা। এই চল-সংজ্ঞা বা ভ্যারিয়েবল-ডেফিনিশনের তিনটে অংশ থাকে। প্রথম অংশটা তার নাম, ধরুন, '*variable_name*'। তার পরে একটা অ্যাসাইনমেন্ট অপারেটর বা ধার্যীকরণ চিহ্ন মানে আমাদের চেনা সমান চিহ্ন '='। তারপর ভ্যারিয়েবলের মান, ধরুন, '*variable_value*'। এখানে আমরা এই '=' চিহ্নকে অ্যাসাইনমেন্ট অপারেটর বলে ডাকছি কেন, সেটা খেয়াল করুন, পরে, শুধু শেল-স্ক্রিপ্টে নয়, যে কোনো কম্পিউটার ভাষাকে বোঝার জন্যেই আপনার এটা কাজে লাগবে। যখনই ভ্যারিয়েবলের নামটা দিলাম, সেই নামে স্মৃতিভূমিকে একখণ্ড জমি চিহ্নিত হল। এবার অ্যাসাইনমেন্ট অপারেটর ব্যাশকে জানাল যে এবার ডানদিকে যে ভ্যারিয়েবলের মান দেওয়া হচ্ছে সেটাকে অ্যাসাইন বা ধার্য করে দিতে হবে, ভরে দিতে হবে ওই জমিটুকুতে। শুধু মাথায় রাখবেন, '=' চিহ্নের আগে এবং পরে কোনো স্পেস দেবেন না। স্পেস দিলে ব্যাশ আর একে অ্যাসাইনমেন্ট অপারেটর বলে চিনতে পারবে না। সংজ্ঞার কাঠামোটা হল — '*variable_name=variable_value*'। এই কাঠামোটার সঙ্গে মিলিয়ে নিন, একটু আগে আমরা যে 'lekho' ভ্যারিয়েবলটা বানিয়েছিলাম, তার সংজ্ঞাটাকে। '*variable_name*' হল জমি, আর '*variable_value*' তার ফলন, এই প্রতিতুলনাটা আমরা আগেই দিয়েছিলাম।

আর এই ব্যাশের বাইরে অন্য যে কোনো ব্যাশ প্রক্রিয়াতেও বা অন্য যে কোনো প্রোগ্রামের কাছেও এই চলটা হাজির করার কৌশলটা তো একটু আগেই দেখলাম আমরা, এক্সপোর্ট করার। '`export` *variable_name*' হল এই রপ্তানি আদেশের কাঠামো। মনে করে দেখুন, কী ভাবে আমরা 'lekho' চলটাকে রপ্তানি করেছিলাম যে কোনো ব্যাশ তথা যে কোনো প্রোগ্রামের কাছে। এই ব্যাশ বা এই প্রোগ্রাম সবগুলোই কিন্তু এই ইউজারের, কারণ এটা আঞ্চলিক চল বা লোকাল ভ্যারিয়েবল। গ্লোবালি মানে গোটা সিস্টেম জুড়ে একটা ভ্যারিয়েবলকে কী করে রুট হাজির করতে পারে, সেটাও জেনেছি আমরা।

এবার, যেই ভ্যারিয়েবলের সংজ্ঞা দিয়ে সেটাকে বানানো এবং তারপর সেটাকে যে কোনো প্রোগ্রামের কাছে রপ্তানি করা হয়ে গেল, তখন এই ইউজারের চালানো যে কোনো প্রোগ্রাম চলটাকে ব্যবহার করতে পারবে। ব্যবহার করতে গেলে তাকে এই ভ্যারিয়েবলের নামটা দিয়ে উল্লেখ করতে হবে, এবং নামের আগে লাগাতে হবে ওই '$' চিহ্নটা, যা দিয়ে আমরা যে কোনো ভ্যারিয়েবলের মানকে বোঝাই। যেমন করে একটু আগে আমরা ব্যবহার করলাম 'lekho' চলটাকে। যে কোনো মুহূর্তে কোনো ভ্যারিয়েবলের জমিতে কোন মান ফলে আছে তা জানার উপায় হল '`echo`'। আদেশ দিন, '`echo` $*variable_name*'। দেখুন স্ক্রিনে তার মানটা, '*variable_value*', ফুটে উঠেছে। যেমন, '`echo $lekho`' আদেশ দিলে আমরা পাব ওই গোটা লাইনটা। দিয়ে দেখুন।

এই ভ্যারিয়েবলগুলোর কনফিগারেশন কী ভাবে আমরা বদলাব তার প্রথমতম কায়দাটা আমরা তো জানিই, কনফিগারেশন ফাইলে কোনো একটা লাইনকে যদি আমরা অকেজো করে দিতে চাই, তাহলে, তার আগে একটা '#' চিহ্ন লাগিয়ে দেওয়া। এটা নানা জায়গাতেই বেশ কাজে আসে। একটা লাইন কী কাজ করছে সেটা বোঝার জন্যে লাইনটার আগে একটা '#' চিহ্ন লাগিয়ে দিলাম। পরে, যদি ফেরত আসার দরকার পড়ে, '#' চিহ্নটাকে উড়িয়ে দিলাম। একে বলে কমেন্টিং আনকমেন্টিং। মন্তব্যীকরণ অমন্তব্যীকরণ। 'মন্তব্য' মানে যা মূল ফাইল নয়, ফাইলেরই কোনো বিষয় নিয়ে কোনো আলোচনা। '#' চিহ্নটা থাকলে ব্যাশ ধরে নেবে, লাইনটা একটা মন্তব্য, আর পড়বেনা। এই মন্তব্যীকরণ আর একটা কারণেও দরকার পড়ে, সেটা হল, একটা কনফিগারেশন ফাইলে বা একটা শেল স্ক্রিপ্টে কোথায় কী করা হচ্ছে সেটার একটা হদিশ দিয়ে দেওয়া। যেমন, '/etc' ডিরেক্টরিতে অনেকগুলো কনফিগারেশন ফাইলেরই কোন অংশটা আসলে কী কাজ করে, সেটা আমরা জানতে পেরেছিলাম ফাইলগুলোর '#' চিহ্ন লাগানো লাইন মানে কমেন্ট লাইনগুলো পড়ে। শুধু এই জন্যে না, যে স্ক্রিপ্টগুলো আপনি নিজে বানাবেন, সেগুলোতেও এটা লাগানো ভারি পুণ্যের কাজ। শুধু অন্যের জন্যে না, নিজের জন্যেও। এটা অবশ্যম্ভাবী যে প্রথম যেদিন আপনি শেল স্ক্রিপ্ট বানাতে শুরু করবেন, প্রথম স্ক্রিপ্ট শেখার ধুমকিতে আপনি যেখান থেকে শিখছেন সেখানে দেওয়া প্রতিটি

নিয়ম নানা ভাবে প্রয়োগ করে নানা স্ক্রিপ্ট লিখবেন, যার বেশির ভাগ কলকব্জাই আপনি মেশিন অফ করে বেরোনোর পরে বেমালুম ভুলে যাবেন, আর পরে কখনো, স্ক্রিপ্টটা দেখে, আপনাকে মাথার চুল ছিড়তে হবে এটা বুঝতে যে স্ক্রিপ্টটা কী করে, বেড়াল তাড়ায় না ঝাড়ে বাঁশ কাটে। তাই, নিজেকে এবং অন্যকে এত কষ্ট দেওয়ার চেয়ে প্রথম থেকেই কমেন্ট লাগানো প্র্যাকটিস করুন। স্টিভ উয়ালিনের সি প্রোগ্রামিং নিয়ে বইটায় সি প্রোগ্রাম লেখার এবং কমেন্ট লেখার ধাঁচ নিয়ে ভারি মজার কিছু আলোচনা আছে, পড়ে দেখতে পারেন, সেটা শেল স্ক্রিপ্ট নিয়েও একই রকম খাটে। সিস্টেমের মধ্যে ব্যাশ ব্যাশ স্ক্রিপ্টিং নিয়ে অর্বুদ অর্বুদ মালপত্তর আছে, নেট থেকেও পাবেন, জিএলটির সিডিতেও আমরা দিই। দরকার হলে যোগাযোগ করবেন। জিএলটির মেল আইডি তো দিয়েছি, আমারটাও দিয়ে দিই, 'paagol@softhome.net'। এই দেড় লাখ শব্দ জুড়ে আমার লেখা পড়তে পড়তে আপনি তো আমাকেও কিছুটা মেপে নিয়েছেন, সেটা বেশ মিলে যাচ্ছেনা? দেখেছেন, স্বতঃব্যাখ্যাশীল ইমেল আইডি মনে রাখার কী সুবিধে। ঠিক একই জিনিষ মনে রাখবেন ভ্যারিয়েবলগুলো নিয়েও, কম টাইপ করার স্বার্থে আপনি খুব ছোট ছোট ভ্যারিয়েবল নাম লাগাতেই পারেন, ঠেলাটা বুঝবেন পরে, যখন কিছুতেই মনে করতে পারবেন না, কোন ভ্যারিয়েবলটা কী। কিন্তু নামটা থেকেই যদি আন্দাজ করা যায়, এই ঝামেলাটাই হয়না। এটাও, শুধু শেল স্ক্রিপ্ট না, যে কোনো প্রোগ্রামিং-এর জন্যেই সত্যি। আর, আরো একবার বলছি, মোটামুটি পরিমাণ কম্পিউটার ব্যবহার করার প্ল্যান থাকলেও, টাচ টাইপিং শিখে নিন, মানে দশ আঙুলে টাইপ করা। প্রথম বছরেই আপনার এতটা সময় বাঁচবে মোট কাজের যে প্রাথমিক পরিশ্রমটা উশুল হয়ে যাবে। গ্নু-লিনাক্সে টাইপ শেখার প্যাকেজ জিটাইপিস্ট (gtypist) — আগেই বলেছি, ভীষণ কাজের।

৭।। বিভিন্ন ব্যাশ ভ্যারিয়েবলের কনফিগারেশন

ব্যাশ ভ্যারিয়েবল কনফিগার করতে নেমে আমাদের প্রথম মুরগী সেই পুরোনো পাপী পথনির্দেশ — 'PATH'। বারবার নানা কাজে নানা প্রসঙ্গে আমরা তাকে টেনে এনেছি, বদলে যেতে দেখেছি, কিন্তু নিজেরা হাতে করে কখনো বদলাইনি। কিন্তু বদলানোর প্রয়োজনটা প্রায়ই হয়, কখনো কখনো সাময়িক, কখনো স্থায়ী রকমে। ধরুন, কোনো একটা ফাইল বা প্রোগ্রাম আপনি আপনার কাজে লাগাতে চাইছেন, যা স্বাভাবিক ভাবে আপনার পথনির্দেশে নেই। একটা উদাহরণ নেওয়া যাক। প্রথমে আমার মেশিনের সুজে সিস্টেমে 'dd' নামের ইউজার হিশেবে আমার পথনির্দেশটা জেনে নেওয়া যাক। এর কমান্ডটা আমরা জানি, '`echo $PATH`'। স্ক্রিনে ফুটে ওঠা এর ফলাফলটা তুলে দেওয়া যাক।

```
/home/dd/bin: /usr/local/bin: /usr/bin: /usr/X11R6/bin: /bin: /usr/games:
/opt/gnome2/bin: /opt/gnome/bin: /opt/kde3/bin: /usr/lib/java/jre/bin:
/opt/gnome/bin
```

এর মধ্যে দেখুন আলাদা আলাদা এগারোটা উপাদান আছে। এগারোটা আলাদা আলাদা পথ। তাদের সমাহারটা হল 'dd' নামক ইউজারের জন্যে গোটা পথনির্দেশ। এই উপাদানগুলো পথনির্দেশের মধ্যে থাকে পরপর, শুধু মধ্যে একটা করে যতিচিহ্ন ':'। কোনো স্পেস থাকেনা। প্রতিটি কোলনের পরে একটা করে স্পেস আমি এখন আনলাম, নইলে যেখানে সেখানে লাইন ভেঙে দিচ্ছে, আর বুঝতেও অসুবিধে হতে পারে একটু অনভ্যস্ত চোখে। প্রথম উপাদানটা দেখুন, 'dd' নামক ইউজারের নিজের হোম হল '/home/dd'। তার মধ্যে একটা বাইনারি ডিরেক্টরি, '/home/dd/bin'। এই ডিরেক্টরিটা কিন্তু এমনিতে থাকেনা হোম ডিরেক্টরিতে। মানে সুজে বা স্ল্যাকওয়ার কোনোটাতেই থাকেনা। কিন্তু পথনির্দেশে থাকা মানে এই ইউজার চাইলেই বানিয়ে নিতে পারে ডিরেক্টরিটা। তার নিজের বানানো কোনো প্রোগ্রাম বা শেল-স্ক্রিপ্ট সে যদি এখানে রাখে সেটা শেল পেয়ে যাবে। আলাদা করে প্রোগ্রামটা কোথায় আছে তার পুরো পথটা দিয়ে দিতে হবেনা। যেমন নয় নম্বর দিনে বারবার 'writefiles' ইত্যাদি যে শেল-স্ক্রিপ্টগুলো আমরা ব্যবহার করেছি, তাদের সবগুলোই রাখা আছে 'dd'-র নিজের বানিয়ে নেওয়া এই '/home/dd/bin' ডিরেক্টরিতে। তাই কমান্ড প্রম্পটে জাস্ট '`writefiles`' টাইপ করে দিলেই স্ক্রিপ্টটা চালাতে পারে ব্যাশ। কিন্তু যদি না-থাকত তখন কী হত?

গুইতে বা গ্রাফিকাল ইউজার ইন্টারফেসে আমি যখন জজজ-এ, মানে ইন্টারনেটে ব্রাউজ বা ঘোরাঘুরি করি, যে ব্রাউজারটা সবচেয়ে বেশি ব্যবহার করি সেটা হল মোজিলা-এক্সএফটি (mozilla-xft)। এই প্যাকেজটা এমনিতে থাকেনা সিস্টেমে, অরিজিত বলেছিল, এটা সুজের জন্যে খুব ভালো কাজ করবে। নেট থেকে নামিয়েছিলাম প্যাকেজটা। টার করা, বিজিপটু কোঁকড়ানো থেকে বড় করে নিয়েছিলাম '/opt' ডিরেক্টরির মধ্যে 'personal' বলে

একটা সাবডিরেক্টরিতে, 'tar xvjf' কমান্ড দিয়ে। এতে '/opt/personal/mozilla' নামে একটা ডিরেক্টরি তৈরি হয়ে গেছিল। '/opt/personal' ডিরেক্টরিতে না-করে সরাসরি '/opt' ডিরেক্টরিতেও করা যেত, কিন্তু সেখানে স্বাভাবিক মোজিলার জন্যে আগে থেকেই একটা '/opt/mozilla' বলে ডিরেক্টরি আছে। মোজিলা-এক্সএফটি আমায় কম্পাইল করতে হয়নি, এটা একটা প্রিকম্পাইলড বাইনারি। মানে এর ভিতরে 'mozilla-xft' নামে একটা বাইনারি বা প্রোগ্রাম ফাইল আগে থেকেই তৈরি হয়ে আছে, যেটা সুজে ৮.২ সিস্টেমের মত করেই তৈরি। এবার সমস্যাটা হল এই যে এই বাইনারিটার ঠিকানাটা দেখুন — '/opt/personal/mozilla/mozill-xft' — এই পথটা কিন্তু আমার '$PATH'-এ নেই। তাহলে আমি কী করব?

একটা আছে ব্রুট ফোর্স বা গাজোয়ারি পন্থা। চালাচ্ছি যখন সরাসরি গোটা পথটা দিয়ে দেওয়া। মানে যখনই মোজিলা-এক্সএফটি চালাচ্ছি, তখনই তার গোটা পথটা দিয়ে দেওয়া — '/opt/personal/mozilla/mozilla-xft'। নইলে ব্যাশ প্রতিবারই একবার করে তার অজ্ঞতা জ্ঞাপন করবে, কমান্ড নট ফাউন্ড। প্রতিবারই এরকম দিতে হবে যতবার আমরা মোজিলা-এক্সএফটি চালাব। এর একটা সহজ উপায় হল, সাময়িক ভাবে 'PATH' ভ্যারিয়েবলটাকে লোকালি কনফিগার করে নেওয়া। এর জন্যে কমান্ড লাগবে দুটো। পরপর।

```
PATH=$PATH:/opt/personal/mozilla
export PATH
```

কমান্ড দুটোর দ্বিতীয়টাকে আমরা ইতিমধ্যেই চিনি। রপ্তানি করে অন্য ব্যাশ তথা প্রোগ্রামের কাছে হাজির করে দেওয়া। প্রথম কমান্ডটাই আসলে চালু 'PATH' ভ্যারিয়েবলটাকে আমাদের প্রয়োজন মোতাবেক বদলে দিচ্ছে। আমরা জানি '=' একটা অ্যাসাইনমেন্ট অপারেটর, একটা বিশেষ মানকে অ্যাসাইন করে দেয়, লাগু করে দেয় একটা বিশেষ ভ্যারিয়েবলের জমিতে। এই '=' চিহ্নের বাঁদিকে থাকে ভ্যারিয়েবলের নাম, ডানদিকে থাকে আমাদের কাঙ্খিত মান। একটু আগে ভ্যারিয়েবলের সংজ্ঞা বা ডেফিনিশনের স্টেটমেন্ট থেকে দেখে নিন। তার মানে, এই চিহ্নের বাঁদিকে আছে 'PATH' নামের ভ্যারিয়েবল, আর ডানদিকে তার যে ভ্যালুটা আমরা চাইছি, মানে নতুন পরিবর্তিত '$PATH'। এবার ডানদিকের পুরোটার মধ্যে দেখুন, দুটো অংশ। তাদের মধ্যে আমাদের এইমাত্র পরিচিত যতিচিহ্ন ':', যে চিহ্নটা, এইমাত্র আমরা দেখেছি, পরপর একাধিক অংশকে একটার সঙ্গে আর একটা যোগ করে যায় '$PATH' মানে 'PATH' ভ্যারিয়েবলের মান বোঝাতে গিয়ে। এখানেও ':' চিহ্নটা ঠিক তাই করছে। নতুন '$PATH'-এ দুটো অংশকে জুড়ে দিচ্ছে। পুরোনো '$PATH', মানে কমান্ড শুরু হওয়ার সময়ে '$PATH' বলতে ব্যাশ যা বোঝে, যাকে সে এখানে উল্লেখ করছে '$PATH' বলে, এবং তার পরে ওই ':' চিহ্ন, সবশেষে সেই অতিরিক্ত পথাংশ যা আমরা চাইছি, মানে '/opt/personal/mozilla'। তার মানে দেখুন, এই কমান্ডটা তার যাত্রা শুরু করছে একটা '$PATH' নিয়ে, এবং শেষ হচ্ছে আর একটা '$PATH' নিয়ে। এই নতুন '$PATH' হল পুরোনো '$PATH' যোগ ওই বাড়তি পথাংশ '/opt/personal/mozilla'।

এবার, নতুন '$PATH' যে সত্যিই বদলে গেছে সেটা আমরা প্রমাণ পাব কী করে? খুব সোজা, আবার সেই ব্যাশকে প্রতিধ্বনি করতে বলা, 'echo $PATH'। এবার যে মানটা পেলাম আমরা সেটায় ওই পথাংশ যোগ হয়ে গেছে।

```
/home/dd/bin: /usr/local/bin: /usr/bin: /usr/X11R6/bin: /bin: /usr/games:
/opt/gnome2/bin:  /opt/gnome/bin:  /opt/kde3/bin:  /usr/lib/java/jre/bin:
/opt/gnome/bin: /opt/personal/mozilla
```

এখন দেখুন, মোট পথাংশ এগারোটার জায়গায় বারোটা হয়ে গেছে। তার বারোতমটা হল আমাদের যোগ করা। ঠিক যেমন আমরা দেখেছিলাম, আগে রয়েছে পুরোনো '$PATH', তার পর এসে গেছে আমাদের যোগ করা অংশটা। এতে লাভটা কী হয়েছে? এখন আমি যেই কমান্ড প্রম্পটে 'mozilla-xft' কমান্ডটা ব্যবহার করব, ব্যাশ আর বলবে না, আমি কী জানি। সে পরপর খুঁজতে খুঁজতে বারো নম্বর অংশটা খোঁজার সময়েই পেয়ে যাবে তার আরব্ধ কমান্ড। মানে আমাদের চাওয়া ওই পথাংশ আমার ব্যাশের '$PATH'-এ এসে গেছে। এই বদলটা কিন্তু সাময়িক। একবার লগ-আউট করে গেলে পরের বার এটা আর পাওয়া যাবেনা। তখন ব্যাশ আবার '$PATH' বলতে প্রথম মানটাই বুঝবে। এবার, আপনি অফিসিয়াল রকমে এই বদলটা যদি পাকাপোক্ত করে নিতে চান, তার উপায় কী?

এরকম কোনো উপায়ের কথা আপনি কি জানেন? যদি না-জানেন, করবেন না, সিম্পল। আমি কী জানি? এটা শুধু ব্যাশবাবু নয়, আমাদের লাগের আঁফা তারিবল সায়মিন্দুর ক্যাচফ্রেজ।

চিকেন-ফ্লু উবে গেছে, এবার একটা মুরগী মানে পথনির্দেশ জবাই করেই থামলে চলবে? মুরগী থেকে মুরগী, সভ্যতা তো এভাবেই এগিয়েছে। সভ্যতার ইতিহাসের দিকে তাকিয়ে দেখুন, ক্রমউন্নতিশীল মুরগী জবাইয়ের কালপথরেখা। আমাদের দ্বিতীয় মুরগী নেওয়া যাক প্রম্পট, কমান্ড প্রম্পট। প্রতিমুহূর্তে যার মুখ আমাদের দেখতে হচ্ছে। মুখ মানে কালো মুখ, তাতে সাদা আখরে ফুটিয়ে তোলা প্রম্পট। যারা পড়ছেন, তার মধ্যে আমার মত বুড়ো যদি কেউ থাকেন, তারা তাদের একসময়কার পুরোনো ডস আর উইনডোজের অভিজ্ঞতায় 'autoexec.bat' নামে ব্যাচ ফাইলের সঙ্গে মিল পাচ্ছেন নিশ্চয়ই। ব্যাচ ফাইলকে একধরনের দুধেভাতে শেল স্ক্রিপ্ট বলতে পারেন। সেখানেও মনে করে দেখুন, প্রথম দুটো বিষয়ই থাকত এই পথ আর প্রম্পট। তার পরেই আসত কটা ফাইল একসঙ্গে খোলা যাবে তার হদিশ। সেসব বিটলেপনা গ্নু-লিনাক্সে নেই। এখানে মেমরি এবং ভারচুয়াল মেমরি ওরকম খাজা রকমে নিয়ন্ত্রণ করা হয়না। প্রথম প্রথম উইনডোজ থেকে গ্নু-লিনাক্সে এসে চমকটা লাগে এখানেই — একসঙ্গে এতকিছু এতগুলো কাজ এত চমৎকার করা যায়, এই মেশিনেই, কই উইনডোজে তো হতনা? এর কারণ, মাল্টিটাস্কিং-টা ডস তথা উইনডোজ সিস্টেমে আরোপিত, প্রক্ষিপ্ত, বাইরে থেকে চাপানো। বহরমপুরের সেই ভদ্রলোকের মত, যিনি মাঝেমাঝেই কয়েকটা করে রবীন্দ্রসঙ্গীত লিখে রেডিওস্টেশনে পাঠিয়ে দিতেন, এবং এদের একচোখোমি দেখে চটে যেতেন, দেখেছ, তিনি মফস্বলে থাকেন বলে তার রবীন্দ্রসঙ্গীত এরা বাজাচ্ছে না। আমাদের চার আর পাঁচ নম্বর দিনের ইউনিক্স বিবর্তনের ইতিহাসের সঙ্গে দু নম্বর দিনের মাল্টিপ্লেক্সিং-এর আলোচনাটাকে আর আট নম্বর দিনের ভার্চুয়াল মেমরি ফাইল সিস্টেমের আলোচনাটাকে মিলিয়ে ভাবুন, তাহলেই বুঝতে পারবেন, ইউনিক্স মাল্টিপ্লেক্সিং-এর মধ্যে দিয়েই তৈরি হয়েছে। এক সঙ্গে অনেক ব্যবহারকারীর অনেক রকম প্রক্রিয়াকে মেলাবেন তিনি মেলাবেন বলেই তৈরি হয়েছিল ইউনিক্স, এবং তারই উত্তরাধিকারে ওপনসোর্স ইউনিক্স মানে গ্নু-লিনাক্স। তাই ঝোড়ো হাওয়া আর পোড়ো বাড়িটার ওই ভাঙা দরজাটা সেখানে জোর করে মেলাতে হয়না, যাতে একটা গান বাজাতে শুরু করলেও ডিফ্র্যাগমেন্টেশন শুধুই বারবার কোঁত পাড়তে থাকে, নতুন করে শুরু হয়, নিজেই গাইতে শুরু করে দেয়, আমারে তুমি অশেষ করেছো। অন্য কিছুর সঙ্গে একই সাথে সিনেমা টিনেমার কথা তো ছেড়েই দিন। যেখানে একটা আপডেট-ডিবি এবং একটা ব্যাকআপ পিছনে ব্যাকগ্রাউন্ড প্রসেসে নিয়ে গ্নু-লিনাক্সে দিব্য আমি, এই গতকালই, একই সঙ্গে পাশাপাশি ম্যাট্রিক্স সিনেমাটা চালিয়ে, পাশে তার চিত্রনাট্যটা খোলা, আর পাশে একটা ইম্যাক্সে তার নোট নিচ্ছিলাম। কিয়ানু রিভস এমন একটা হেঁচকি অব্দি তুলল না যা চিত্রনাট্যে নেই, মাইরি, মিলিয়ে বলছি। চিত্রনাট্যটা নেট থেকে নামানো, অরিজিনাল।

হ্যাঁ, প্রম্পট। এই প্রম্পট করার কেউ নেই বলেই, মূল পার্ট ভুলে গিয়ে মাঝে মাঝেই বাজে বকতে থাকছি। যাকগে এখন আর ভেবে লাভ নেই, বইই শেষ হয়ে এল। এক লাখ পঁয়তাল্লিশ হাজার শব্দ নামিয়ে দিয়েছি, আর হাজার পাঁচেকের মধ্যেই ছুটি, সেই নভেম্বরের শুরু থেকে আজ মার্চের তিন, আমাদের লেঠেল বংশের জিনে দম প্রচুর, তাও, সেটাও ফুরিয়ে আসছে এবার। আর আপনার কথা ভেবে, সত্যি বলছি, আমার নিজেরি কান্না পাচ্ছে। যাইহোক, আপনার প্রম্পটের একটা আকার আছে। যেখানে আপনি কমান্ড দিচ্ছেন। আমরা যেমন পাঁচ নম্বর দিনে ম্যানড্রেকের কমান্ড প্রম্পটটা দেখিয়েছিলাম। ডিস্ট্রো থেকে ডিস্ট্রোয় সেটা একটু আধটু তফাত হয়। ডিফল্ট সেটিং-এ। সেটাকে নিজের মত করে বদলেও নেওয়া যায়। এই কাজে যে ভ্যারিয়েবলটা ব্যবহার করা হয় তার নাম 'PS1'। 'PATH'-এর মত আর একটা এনভায়রনমেন্ট ভ্যারিয়েবল। তবে একেও আপনি লোকালি নিজের মত করে বদলে নিতে পারেন। সিস্টেমে অন্য ইউজারের প্রম্পট তাতে বদলাবে না, শুধু আপনারটা বদলে যাবে। যেমন, সুজে সিস্টেমে প্রম্পটটা দেখুন, এটা ইউজার 'dd'-র জন্যে। অন্য ইউজারে ওই অংশটা বদলে যাবে।

```
dd@mahammad:~/Documents/lesson>
```

এবার দেখা যাক 'PS1' ভ্যারিয়েবলের কী মান কমান্ড প্রম্পটটাকে এই আকার দিচ্ছে। তার জন্যে আমরা কমান্ডটা জানি '`echo $PS1`'। এই কমান্ড দেওয়ায় ব্যাশ স্ক্রিনে ফুটিয়ে তুলল নিচের এই শব্দবন্ধ। হঠাৎ করে দেখলে মনে হতেই পারে, ব্যাশ আমায় হিব্রু ভাষায় গালাগাল দিচ্ছে। এত খাটাই যে।

```
\u@\h:\w>
```

এই '`$PS1`'-এর শেষ '>' চিহ্নটা কমান্ড প্রম্পটের সঙ্গে মিলিয়ে নিন। দুটোতেই এক। এবার দেখুন, আরো একটা চিহ্ন একদম এক। সেটা হল '@'। মানে অ্যাকাউন্টান্সিতে যা ছিল 'অ্যাট-দি-রেট-অফ', এখন নেট পরিভাষায় 'অ্যাট'। এখানে ঠিক 'অ্যাট' এই অর্থেই ব্যবহার হচ্ছে। এবং নেটের লিংকগুলো, ওয়েবসাইটের মধ্যেকার বিভিন্ন ঠিকানা এমনকি উইনডোজেও দেখায়, খেয়াল করবেন, '/' যতিচিহ্ন ব্যবহার করে, উইনডোজের পথনির্দেশের স্বাভাবিক যতিচিহ্ন কিন্তু '\', ছয় নম্বর দিনের সেকশন ৪ থেকে দেখুন। আসলে এই নেট ওয়েব তার বিভিন্ন ক্রিয়াকলাপের ব্যাকরণ — এই গোটাটাই গজিয়ে উঠেছে ইউনিক্স জগত থেকে। সেই ব্যাকরণ মেনেই, নিজের সিস্টেমে যে পথনির্দেশই দেখা, ওয়েবসাইটের বেলায় উইনডোজকেও সেটা মানতে হয়। সেটা বোঝা যাবে এর ঠিক আগের আর পরের '`\u`' আর '`\h`' অংশদুটোর মানে জানলে। ম্যানুয়াল দেখে তো এখনি জেনে যেতে পারেন, সেটা হল জানার বোঝার সাহেব রকম, সব কিছু জেনেই যাদের শিখতে এবং বুঝতে হয়। আমরা তো বহুসময় বইপত্তরই পাই না, তাই, বাধ্যতই, নিজেকেই চেষ্টা করতে হয়। আমরা জানি বাঁদিকের অংশটা যে ইউজারের ব্যাশ তার নাম, সেখান থেকে আন্দাজ করা যায় '`\u`' মানে নিশ্চয়ই ইউজারের নাম। এবার বাকি রইল '`\h`'। প্রম্পটে যার আকার হল 'mahammad'। নয় নম্বর দিনে, মনে করে দেখুন তো, এই নামটা পেয়েছিলেন? পেয়েছিলেন হোস্টনেম কনফিগারেশনের ফাইলের সূত্রে। এবার হোস্টনেমের এইচ আর এই এইচ, আর যায় কোথায়? নিশ্চয়ই এটা হোস্টনেম। আর তারপরে একটা '~' চিহ্ন। এটা নিশ্চিতভাবেই হোম-ডিরেক্টরি, ব্যাশ '~' বলতেই চালু ইউজারের হোম ডিরেক্টরি বোঝে। এবার মিলিয়ে দেখুন। আবার সেই ব্যাশের ম্যানুয়াল। একটা আস্ত সেকশন আছে দেখুন, 'PROMPTING'। সেই সেকশন থেকে এই রকম কয়েকটা চিহ্নের মানে তুলে দেওয়া যাক।

```
\d      the date in "Weekday Month Date" format (e.g., "Tue May 26")
\h      the hostname up to the first `.'
\t      the current time in 24-hour HH:MM:SS format
\u      the username of the current user
\w      the current working directory
```

মিলিয়ে দেখুন, একটু আমাদের প্রম্পটের আকারের সঙ্গে 'PS1' ভ্যারিয়েবলের যে আকারটা আমরা পেলাম, '`\u@\h:\w>`', সেটাকে বুঝতে পারছেন কিনা। এবার, যেই আমরা ফরমুলাগুলো দেখে ফেললাম, আমাদের প্রম্পট বদলানো আটকায় কার পিতাশ্রী? কমান্ড দেওয়া যাক, '`PS1="\u@\h<\w>\t>"`', যাতে ফরমুলা মোতাবেক আগের প্রম্পটের উপাদানগুলোর এখন সময়টাও দেখায়। ঠিক তাই হল। প্রম্পটের আকার হয়ে গেল —

```
dd@mahammad<~>23:07:15>
```

এবার চালু কাজের ডিরেক্টরিটাকে আমরা দুপাশে দুটো '<' আর '>' চিহ্নের মধ্যে দিতে বলেছিলাম, ঠিক তাই করেছে ব্যাশ। আর তারপর দিতে বলেছিলাম '`\t`', মানে সময়, চবিবশ ঘন্টার ফরম্যাটে, মানে দুপুর একটা যেখানে তেরোটা, ট্রেনের সময়ের মত। এরকম আর যা যা আপনার মন চায় দিতে পারেন। অজস্র অপশন আছে ব্যাশের ম্যানুয়ালে। আমি ঠিক দরকার যে কটা তার বাইরে আর একটাও দিইনি। মানে কয়েকটা দিয়ে ফেলেছিলাম, খেয়াল করে আবার উড়িয়ে দিলাম। এখান থেকেই পড়ে করে নেবেন, সে চলবেনা। নিজে বার করে নিজে পড়ুন। প্রম্পটের এই অস্থায়ী বদলকে স্থায়ী করবেন কী করে? ঠিক 'PATH' ভ্যারিয়েবলের মত, 'PS1' ভ্যারিয়েবলও এই ইউজারের ব্যাশে স্থায়ী হবে তখনই যখন একে লোকাল কনফিগারেশনে তুলে দেবেন। খেয়াল রাখুন, এটা গ্লোবাল এনভায়রনমেন্ট ভ্যারিয়েবল তখনই হয়ে উঠবে যখন এর ঠাঁই হবে গ্লোবাল কনফিগারেশন ফাইলে।

৮।। এক কুচো ব্যাশ ফাংশন

আজ দোলের সকাল, একটু ফাংশন না-করলে চলে? পরিচালকদের প্রতি প্রযোজকদের লিংগোয় বলতে গেলে, 'এখানে ভ্যারিয়েবলটা আর একটু এক্সপোজ করা চাইছিল না?', ঠিকই, কিন্তু, বেশি এক্সপোজ করার রিস্ক নিলাম না, বসন্তোসবের দিন বলে কথা, প্রাচীন উন্মুক্ত ভারতে যেদিন নর্মদায় যুবকযুবতীরা জন্মদিনের পোষাকে সাঁতার প্রতিযোগিতায় নামত। নর্মদার জল হুগলী অব্দি যে গড়াতে চাইবে না, কে বলতে পারে? তার চেয়ে একটু ফাংশন টাংশন করাই ভালো, লুচিশীল চল্লিশ কোমরের বাঙালির রুচিটুচিও রইল, আবার আবির মাখানোর কন্ট্রোলড বেলেল্লাও হল।

আগেই বলেছি আমরা এই ফাংশনগুলো থাকে '/etc/bashrc' বা '~/.bashrc' ফাইলে। গ্লোবাল ফাংশন এবং লোকাল ফাংশন। ফাংশন কী তাই নিয়ে একাধিকবার কথা হয়েছে আমাদের, কিছু নির্দিষ্ট কাজের একত্র একটা সমাহার। যে কাজগুলো প্রায়ই একসঙ্গে করতে হয়, তাদের একটা সমাহার। সমাহারটা যাতে যে কোনো প্রোগ্রাম কাজে লাগাতে পারে তার দরকার মত, তাই কিছু নির্দিষ্ট ফাংশনকে ব্যাশের কনফিগারেশন ফাইলেই দিয়ে দেওয়া হচ্ছে। ফাংশন এবং লাইব্রেরি নিয়ে আলোচনা মনে আছে? এই ব্যাশ কনফিগারেশন ফাইলটা সেই অর্থে একটা আনবিক লাইব্রেরি বলে ভাবতে পারেন।

ধরুন, আপনার লেখা প্রবন্ধগুলোকে আপনি প্রায়ই 'bzip2' করে '.bz2' এক্সটেনশনের ফাইল করেন। করতেই হয়, কারণ, পাঠক পাওয়ার আগ্রাসী আকাঙ্খায় আপনাকে সেগুলো ইমেল করে পাঠাতে হয়, এবং একই সঙ্গে, খুবই অবিশ্বাস্য যদিও, আপনাকে খেয়ে পরে বাঁচতে হয়, এবং '.bz2' মানে মূল ফাইলটার সাইজ, আরো যদি টেক্সট ফাইল হয়, মোটামুটি দশ থেকে কুড়ি শতাংশ, তার মানে আপনার অনলাইন থাকার টাইমও। আর যদি আপনি লেখেন, এই আকাঙ্খাটা কতটা বিধবংসী হতে পারে, তার প্রমাণের সিরিজে শেষতম হল সঙ্কর্ষণের মন্তব্য, 'তুমি তো মাইরি ফিডব্যাক চেয়ে খুঁচিয়ে গ্যাংগ্রিন করে দিলে'। গত দুই দশকের আমার সংগ্রহ থেকে দুটি মাত্র সূক্তিরত্নাবলী আপনাদের শুনিয়ে রাখি। পিউ একবার বলেছিল, 'দীপঙ্করদা, এই পরিমাণ লিখলে কিন্তু এখন থেকে শোনার চার্জ লাগবে পাতা প্রতি দশ টাকা'। আর শ্রেষ্ঠতমটা হল বিবেকের, বিবেকদংশনের মত, আমি তখন 'তপন বিশ্বাসের খিদের বত্রিশ ঘন্টা' বলে একটা উপন্যাস লিখছি, '৯০-এর এপ্রিল থেকে সেপ্টেম্বর, আমার কাছে পড়তে এসে রোজ দেখত, খেয়ে না-খেয়ে লিখে যাচ্ছি, দু-একদিন একটু আধটু উল্টে পাল্টে দেখেছেও, একদিন খুব উৎসাহভরে বলল, 'শোনো, তোমার এই উপন্যাসটা শেষ হলে আমায় একটু দিও তো, একটা লোককে হেভি বোর করার ইচ্ছে আছে'। হায়। হায় হায়।

এবার একটা ফাংশন আপনি বানিয়ে রাখতে পারেন আপনার '~/.bashrc' ফাইলে যাতে এই গোটা কাজটাই এককথায় হয়ে যায়, আলাদা করে ধরে ধরে কমান্ড দিয়ে কাজটা না-করতে হয়। এই অংশটা যোগ করে দিন আপনার '~/.bashrc' ফাইলে। যোগ করার পর একবার '<Ctrl><D>' বা '`exit`' করে বেরিয়ে যান, দেখবেন লগ-ইনের আমন্ত্রণ জ্বলজ্বল করছে কমান্ড প্রম্পটে, আর একবার লগ-ইন করে নিন। তার আগে অব্দি ব্যাশ ফাংশনটা পাবে না, কারণ নতুন '~/.bashrc' সে তখনো পড়েনি। ব্যাশ ম্যানুয়াল পড়ে দেখুন, কী একটা অন্য উপায় আছে লগ-আউট না করেই কনফিগারেশন ফাইল ব্যাশকে দিয়ে পড়িয়ে নেওয়ার, আমার মনে পড়ছে না।

```
packall()
{
for file in *
do bzip2 -9 $file
done
}
```

এতে কী হল, বলুন তো? এখন থেকে যে কোনো ডিরেক্টরির মধ্যে দাঁড়িয়ে, আপনি '`packall`' কমান্ড দিলেই গোটা ডিরেক্টরি জুড়ে আপনার প্রতিটা প্রবন্ধ একটা করে আলাদা আলাদা একটা '.bz2' ফাইল হয়ে গেছে। পাঠান না, যত খুশি, যাকে পাঠাবেন, ঠেহায় কেডা। এবং এটা দিয়ে আপনি আপনার ডিস্কভূমিও বাঁচাতে পারেন, গোটা ডিরেক্টরির সাইজও অনেক ছোট করে দেয়, কারণ '`bzip2`' করার পরে মূল ফাইলগুলো উড়িয়ে দিয়েছে। আর '`bzip2`' করেছে অপশন '`-9`' দিয়ে, মানে সর্বোচ্চ সম্ভব কুঁকড়ে দিয়েছে। আগের শেলস্ক্রিপ্টের ব্যাখ্যাগুলো যদি মন দিয়ে পড়ে থাকেন, তাহলে এটা বুঝতে আপনার কোনো অসুবিধে হচ্ছেনা, আর যদি না-পড়ে থাকেন, তাহলে যান দোল খেলুন। আর আপনি যদি কালেক্টিভ ফার্মিং-এ বিশ্বাসী না হন, একটা প্রবন্ধ শেষ করেন, একটা ফাইল পাঠান, তাহলে আপনার '~/.bashrc' ফাইলে যোগ করবেন নিচের লাইনগুলো। এটা 'packone()' নামে একটা ফাংশন। আগের 'packall()' ফাংশনটার সঙ্গে এটার তফাত এই যে এটা আর ডিরেক্টরি ধরে কাজ করেনা, কাজ করে ফাইল ধরে। আগেরটা যেমন একটা ডিরেক্টরিতে দাঁড়িয়ে '`packall`' কমান্ড দিলেই চলত, পরের 'packone()' ফাংশনটার জন্যে প্রথমে '`packone`', তার পরে একটা ফাইলনাম যোগ করে দিয়ে কমান্ডটা সম্পূর্ণ করতে হবে। তখন ব্যাশ সেই ফাইলটাকে কুঁকড়ে একটা '.bz2' ফাইল বানিয়ে দেবে। একটা জিনিষ খেয়াল করবেন প্রত্যেকটা ফাংশন হেডিং-

এ, মানে ফাংশন যেখানে তার নাম দিয়ে শুরু হচ্ছে, আমরা ফাংশনের নামের পরেই একটা জোড়া ব্রাকেট ' () ' দিচ্ছি। এটাই নিয়ম। এটা দিয়েই ব্যাশ একটা ফাংশনকে চেনে। ফাংশনটা ব্যবহার করার সময় কিন্তু ব্রাকেটজোড়া আর ব্যবহার করতে হচ্ছে না।

```
packone()
{
bzip2 -9 $1
}
```

এখানে একটা জিনিষ এখনো আমাদের অজানা, '$1'। একটু বাদেই আমরা দেখব, এর মানে, 'packone' কমান্ডটার পরে যে ফাইলটার নাম আপনি দিচ্ছেন। এটা ব্যাশের আর একটা রেডিমেড ফরমুলা। ব্যাশ প্রম্পটে কোনো কমান্ড দেওয়া মানেই তার পরের শব্দটা হল '$1'। ধরুন, এইমাত্র আপনি শেষ করেছেন 'essay.4.text' ফাইলটা। আপনি এবার কমান্ড দেবেন 'packone essay.4.text'। কারণ, এখন একটা ফাইলের নাম দিতে হবে। এতে ব্যাশ তার চেনা ফাংশন 'packone' কাজে লাগিয়ে সঙ্গে সঙ্গে 'essay.4.text.bz2' নামে একটা ফাইল বানিয়ে দেবে। কারণ, দেখুন, ও তুলেছে 'essay.4.text' ফাইলটা, তাকে 'bzip2' করে বানিয়েছে 'essay.4.text.bz2'। যদি আমরা এখানে শুধু 'packone' কমান্ড দিই, কোনো ফাইলনাম না-দিয়ে, ব্যাশ আমাদের জানাবে —

```
bzip2: I won't write compressed data to a terminal.
bzip2: For help, type: `bzip2 --help'.
```

সিস্টেম বোঝার চেষ্টায় একটা খুব ভালো ব্যায়াম এই গন্ডগোলের খবর বা এরর মেসেজগুলো পড়ার এবং বোঝার চেষ্টা করা। দেখুন তো, ব্যাশ যেরকম বলেছে, 'bzip2 --help' করে বা তার বড়ভাই 'man bzip2' করে আপনি এই এরর মেসেজটা কতটা বুঝতে পারেন। এবার, 'packone()' আর 'packall()' ফাংশনদুটোর সঙ্গে একটা এর বিপরীত ফাংশনও আপনি বানিয়ে নিতে পারেন। যে কোঁকড়ানো ফাইলগুলোকে বড় করবে। ধরুন তার নাম দিলেন 'unpack()'। তখন আপনি এই রকম কিছু লাইন যোগ করবেন আপনার '~/.bashrc' ফাইলে।

```
unpack()
{
for pack in *.bz2
do bunzip2 $pack
done
}
```

এই ফাংশনটায় দেখুন ভ্যারিয়েবলটার নাম দেওয়া হয়েছে 'pack', আগের 'packall()' ফাংশনটায় যেমন দেওয়া হয়েছিল 'file', এটা গোটাটাই নিজের খুশি মত, ঠিক ফাংশনের নাম যেমন। এই 'unpack()' ফাংশনটা ঠিক 'packall()' ফাংশনের মত, কোনো ফাইলনাম দিতে হয়না, গোটা ডিরেক্টরির ফাইলতালিকা ধরে কাজ করে। এই যে লেখা 'for pack in *.bz2', এর মানে এই ডিরেক্টরিতে দাঁড়িয়ে 'ls *.bz2' কমান্ড দিলে আমরা যে তালিকাটা পেতাম, মানে গোটা ডিরেক্টরিতে যাবতীয় '.bz2' ফাইলের তালিকা, সেই তালিকাটা পড়ে নিচ্ছে ব্যাশ। তারপর তার প্রথম ফাইলনাম হচ্ছে 'pack' ভ্যারিয়েবলের প্রথম মান, দ্বিতীয় ফাইলনাম হচ্ছে দ্বিতীয় মান, এই ভাবে এগিয়ে যাচ্ছে। আমাদের এই ব্যাশ ফাংশন কাণ্ড শেষ হওয়ার আগে একটা কথা মনে করিয়ে দিই, এই ফাংশনগুলো এক ভাবে দেখলে সম্পূর্ণ অর্থহীন, অ্যালিয়াস দিয়ে খুব সহজেই আপনি করে নিতে পারতেন এটা। কিন্তু, আগেই তো বলেছি, এই লেখাটা আপনাকে সাহায্য করার চেষ্টা করছে আপনি কী ভাবে নিজেকে শেখাবেন সেটা বুঝে ওঠার কাজে। তার চেয়ে বেশি এগোতে হবে আপনাকে নিজেকেই।

৯।। ব্যাশ একটা স্ক্রিপ্টিং ল্যাংগুয়েজ

বারবার নানা উদাহরণ দিয়ে নয় নম্বর দিন থেকে আমরা দেখে আসছি, ব্যাশে কমান্ড প্রম্পটে যে সব কমান্ড দিয়ে আমরা নানা কাজ করছি, সেই কমান্ডগুলোকে একসঙ্গে গেঁথে দেওয়া যায়। এইমাত্র যে ফাংশন নিয়ে আমরা আলোচনা করলাম, সেই ফাংশনও একটা গ্রথিত কমান্ডসমাহার। কিন্তু তার সঙ্গে ব্যাশ স্ক্রিপ্টের মূল পার্থক্য হল স্বাধীন স্বতন্ত্র অস্তিত্বের। ফাংশন হল একটা কমান্ডগুচ্ছ যে স্বনির্ভর এবং স্বতন্ত্র নয়, তাকে ডেকে আনে অন্য কোনো প্রোগ্রাম বা শেল। যেমন এইমাত্র আমরা দেখলাম, কমান্ড লাইন থেকে, অর্থাৎ ব্যাশ দিয়ে তাদের কী ভাবে ডাকতে

হয়। শুধু ব্যাশ কমান্ড প্রম্পট নয়, এই ফাংশনগুলোকে ডেকে নিতে পারে কোনো ব্যাশ স্ক্রিপ্টও। কিন্তু, যেই ডাকুক, তার চলার ভিতর থেকে গজিয়ে ওঠে ফাংশনটার চলা, ফাংশনটা নিজে নিজে চলেনা। তাকে নিজে নিজে চালাতে গেলে কী করতে হয় সেই কায়দাটা আমরা আগেই শিখেছি। যেমন আগের সেকশনে যে ফাংশনগুলো দেখলাম তার যে কোনো একটা, ধরুন 'unpack()' ফাংশনের গোটা কমান্ডমালাটা আমরা যদি একটা ফাইলে তুলে নিই, এবং তার আগে সেই '`#!/bin/bash`' লাইনটা যোগ করে নিই, এবং গোটাটাকে একটা স্বতন্ত্র ফাইল 'unpack.sh' হিশেবে সেভ করে, '`chmod`' কমান্ড দিয়ে তাকে এক্সিকিউটেবল বা চালনীয় করে নিই, তখন এই 'unpack.sh' ফাইলটা একটা ব্যাশ স্ক্রিপ্ট হয়ে গেল। তার কাজ হল হুবহু ওই 'unpack()' ফাংশনটার সঙ্গে একই, কিন্তু সে এখন স্বতন্ত্র স্বাধীন একটা ব্যাশ স্ক্রিপ্ট। সেই ফাইলটায় এই লাইনগুলো থাকতে পারে। দেখুন, এখন আমরা দুটো লাইন যোগ করেছি, এক হল এর নাম ধাম, অন্যটা এর চাকরিবাকরি। এটা একটা প্রথা, খুব কাজের প্রথা। আগেই বলেছি। আর ফাইলনামে '.sh' এক্সটেনশনটাও একটা প্রথা, চলে আসছে, একটা অভ্যস্ততা। তবে যে নামই দিন ব্যাশ স্ক্রিপ্ট হিশেবে কাজ করতে তার কোনো অসুবিধে হবেনা।

```
#!/bin/bash

# Bashscript named 'unpack.sh'
# Expands all 'bz2 ' files in the directory

for pack in *.bz2
    do bunzip2 $pack
done
```

এবার এটাকে যদি কোনো ইউজার তার '/bin' ডিরেক্টরিতে রেখে দেয় তাহলে সেটা ব্যাশ শেলের পথনির্দেশে চলে এল, এবং যে কোনো ডিরেক্টরি থেকেই কমান্ড প্রম্পটে এই '`unpack.sh`' কমান্ডটা দিলেই যাবতীয় কোঁকড়ানো ফাইল বড় করার এই কাজটা করে দেবে স্ক্রিপ্টটা। শুধু '/bin' ডিরেক্টরি বাদে, সেখানে কোনো '.bz2' ফাইল বড় করতে চাইলে আপনাকে কমান্ড দিতে হবে '`./unpack.sh`'। ডট বা বিন্দু বা '.' মানে এখানে বর্তমান বা কারেন্ট ডিরেক্টরি। গ্নু-লিনাক্সে কারেন্ট ডিরেক্টরি শেলের পথনির্দেশে থাকেনা, সিস্টেমের নিরাপত্তার সঙ্গে জড়িত তার গল্পটা বোধহয় আগেই করেছি, যদি না-করে থাকি নিজে খুঁজে পড়ে নিন।

ব্যাশ শেল তাই শুধু একটা কমান্ড প্রম্পট না, নিজেই একটা প্রোগ্রাম করার ভাষা, যাকে বলে স্ক্রিপ্টিং ল্যাংগুয়েজ। একে কিন্তু প্রোগ্রামিং ল্যাংগুয়েজ বলেনা, বলে স্ক্রিপ্টিং, খেয়াল করুন। সি ইত্যাদি প্রোগ্রামিং ভাষায় লেখা কোড চালাতে গেলে আগে কম্পাইল করে নিতে হয় কোডটা। গ্নু-লিনাক্সের কম্পাইলার জিসিসি, গোটা গ্নু-লিনাক্স সিস্টেমটাই যার উপর দাঁড়িয়ে আছে, সেটা আপনার গ্নু-লিনাক্স মেশিনে এমনিতেই আছে। সেই কম্পাইলার দিয়ে আপনি যখন সি ভাষায় লেখা প্রোগ্রাম কম্পাইল করেন, জিসিসি সেই মানববোধ্য কোডটাকে পড়ে মেশিনবোধ্য চালনীয় ফাইল বা এক্সিকিউটেবল ফাইল করে দেয়, আগেই বহুবার বলেছি। এই গোটা কাজটাকেই চালু অর্থে প্রোগ্রামিং বলা হয়। সি বা অন্য যে কোনো হাইলেভেল মানববোধ্য ভাষার নিয়মকানুন মেনে কিছু কোড লিখলেন, কোডিং, তাকে কম্পাইল করলেন, কম্পাইলিং, কম্পাইল করতে গিয়ে যে যে জায়গায় আটকাল বা পরে চলার সময়ে বুঝলেন যে যে জায়গায় ত্রুটি রয়ে গেছে, বা আর একটু ভালো করে করা যেত — সেগুলো সংশোধন করলেন বা পোকা বাছলেন, ডিবাগিং, এবং সবশেষে পোকা ছাড়ানো মনোমত কোড থেকে কম্পাইল করে চূড়ান্ত চালনীয় ফাইল বা এক্সিকিউটেবল ফাইল বানালেন — এই গোটাটাকে মিলিয়েই হল প্রোগ্রামিং।

এর সঙ্গে স্ক্রিপ্টিং-এর কিছু মৌলিক তফাত আছে। স্ক্রিপ্টিং বলতে আমরা বোঝাচ্ছি, কিছু ব্যাশ বা অন্য কোনো শেলের কমান্ডকে নেওয়া, তাদের একত্রে সমাহৃত করা একটা ফাইলে, স্ক্রিপ্ট লেখার নিয়মকানুন মেনে, যেমন গোড়াতেই '`#!/bin/bash`' ইত্যাদি ওই বিশেষ লাইনটা লাগিয়ে নেওয়া একটা প্রাথমিকতম নিয়ম, আর '`chmod`' দিয়ে এক্সিকিউটেবল করে নেওয়া আর একটা, তারপর তাকে চালানোর চেষ্টা করা এবং যথারীতি পোকা ছাড়ানো, আর সব শেষে চূড়ান্ত ফাইলটা চালানো — একে বলে স্ক্রিপ্টিং। এখানে একটা মজা আছে, খেয়াল করুন, আমরা তো আমাদের এক নম্বর দিনেই কম্পিউটারকে, একটু আদর করেই, গাধা বলে ডেকেছিলাম, এবং সেই সূত্রে বলেছিলাম, মেশিন জিনিষটা আসলে একটা দামী গাম্বাট। অত্যন্ত নিরেট। নিজে কিছুই বোঝেনা। তাকে প্রতিটি আদেশ তার মত

করে বুঝিয়ে দিতে হয়। কম্পাইলারটা আসলে যা করে। আমাদের প্রখর বুদ্ধি আর মেশিনের জটিল এবং মহার্ঘ নির্বুদ্ধিতার মধ্যে দোভাষির কাজ করে। কিন্তু স্ক্রিপ্টের বেলায় তো কোনো কম্পাইলার লাগছে না, আমাদের চেনা পুরোনো মেশিনটাকে এমন বৈপ্লবিক ব্রেনোলিয়া খাওয়াল কে? আর কে? সেই ব্যাশ। ব্যাশই এখানে দোভাষি। কিন্তু তার অনুবাদ-কাঠামোটা কম্পাইলারের অনুবাদ প্রক্রিয়া থেকে আলাদা। একে বলে ইন্টারপ্রিটেশন, এবং যে প্রোগ্রাম এই অন্যরকম অনুবাদ মানে ইন্টারপ্রিটেশনের কাজ করে তাকে বলে ইন্টারপ্রিটার। এখানে ব্যাশ হল সেই ইন্টারপ্রিটার।

প্রোগ্রামিং ভাষা আর স্ক্রিপ্টিং ল্যাংগুয়েজ এর মধ্যে প্রাথমিক পার্থক্যটা এই যে প্রোগ্রামিং ভাষা স্বভাবত অনেক বেশি শক্তিশালী। এবং কাজ করতে পারে অনেক দ্রুত। প্রোগ্রামিং ভাষার উদাহরণ সি, ফোর্ট্রান, সিপ্লাসপ্লাস, জাভা ইত্যাদি। এই ভাষাগুলোয় লেখা সোর্স কোড কম্পাইল করে বাইনারি বানানো এবং একটা অপারেটিং সিস্টেম থেকে অন্য অপারেটিং সিস্টেমে তাদের স্থানান্তরযোগ্যতা বা পোর্টেবিলিটি নিয়ে আমরা আলোচনা করেছি চার পাঁচ ছয় এবং আট নম্বর দিনে। একই সোর্স কোড থেকে একটা অপারেটিং সিস্টেমের জন্যে বানানো এক্সিকিউটেবল অন্য অপারেটিং সিস্টেমে চলে না। আবার ওই অপারেটিং সিস্টেমের মত করে তাকে কম্পাইল করে নিতে হয়।

একটা স্ক্রিপ্টিং ভাষাও সেই অর্থে শুরু করে সোর্স কোডে। আমাদের বানানো স্ক্রিপ্টগুলোয় আমরা যে আদেশ বা কমান্ড সমাহার টাইপ করে তুলে দিচ্ছি সেটাই ওই স্ক্রিপ্টের সোর্স কোড। কম্পাইলার গোটা সোর্স কোডকে একত্রে চটকায় মানে প্রসেস করে। কিন্তু ইন্টারপ্রিটার হল শোভন রুচিশীল, সে আদেশ বাই আদেশ এগোয়। একটা ধরো পড়ো করো, তারপর আর একটা। কিন্তু এই রকম ভেঙে ভেঙে পড়ার কারণে তার গতিটাও কম্পাইলড ভাষাগুলোর তুলনায় অনেক কমে যায়। কিন্তু সুবিধে আছে এই যে এই কমান্ড গুলোকে বুঝতে পারে এমন কোনো শেল যদি একটা অপারেটিং সিস্টেমে থাকে, সেই অপারেটিং সিস্টেমেই এই স্ক্রিপ্টগুলো চলবে। যেমন ধরুন আজকের আলোচনার এক নম্বর সেকশনেই আমরা 'mplayer' নামে প্যাকেজের সোর্স কোড থেকে বাইনারি কম্পাইল করে নেওয়ার কথা বলেছিলাম। প্রথমে '`configure`' কমান্ড দিয়েছিলাম। এই 'configure' একটা প্রোগ্রাম যা গোটা মেশিনের অপারেটিং সিস্টেমের কনফিগারেশনের খুঁটিনাটি দেখে নেয় এবং সেই মোতাবেক একটা ফাইল তৈরি করে যা পরে জিসিসি নামে কম্পাইলারের কাজে লাগবে কম্পাইল করার সময়। ওই যে '`make`' এবং '`make install`' কমান্ড দুটো — ওরাও ডেকে এনেছিল জিসিসি নামের কম্পাইলারকে, ঠিক কী কী ভাবে কম্পাইল এবং ইনস্টল করতে হবে এইমাত্র কনফিগার করা তথ্য মেনে — সেই কথা বলে দিয়েছিল। 'make' হল গ্নু-র একটা উপযোগিতা প্যাকেজ, রিচার্ড স্টলম্যান এবং রোলাঁ ম্যাকগ্রার বানানো। একটু মেক-এর খেজুরগাছে চড়ে আসতে পারেন, তাতে আপনার উপকারই হবে। আর ভীষণ ভালো একটা বই, 'লিনাক্স ডিভেলপমেন্ট প্ল্যাটফর্ম', রফিক উর রহমান এবং ক্রিস্টফার পলের লেখা, মোটের উপর গ্নু-লিনাক্স সিস্টেমে যে কোনো প্রোগ্রামিং করা নিয়ে, নেটে ফ্রিতে ডাউনলোড করা যায়, সেটা পড়ে নিতে পারেন, কোনো রকমের প্রোগ্রামিং নিয়ে আপনার যদি মিনিমাম আগ্রহ থাকে। তাতে মেক নিয়ে বিরাট একটা চ্যাপ্টার আছে। মেক কাজ করে মেকফাইল (makefile) দিয়ে, যে মেকফাইল হল ওই এমপ্লেয়ার কোড ডিরেক্টরিতে রাখা আছে — একটা গাইড, কী ভাবে কম্পাইল করতে হবে। সেই ফাইল অনুযায়ী জিসিসি কম্পাইল করে একটা বাইনারি বানায়, যার নাম 'mplayer', এবং তাকে '/usr/local/bin' ডিরেক্টরিতে রেখে দেয়। এই ডিরেক্টরি ব্যাশ শেলের স্বাভাবিক পথনির্দেশের মধ্যে পড়ে, তাই এর পর থেকে, যে ডিরেক্টরিতে বসেই আমি '`mplayer`' কমান্ড দিই-না কেন, আমি সিনেমা দেখতে পাব বা গান শুনতে পাব। এই গোটা কাজটাই আমি করেছিলাম আমার মেশিনের সুজে ৮.২ সিস্টেমে। তার মানে এর অন্তর্বর্তী কনফিগারেশনটা এই সিস্টেমের। এবার এটা লিখতে লিখতেই শখ হল, কী হয় দেখি, আমি সুজে সিস্টেমের ওই '/usr/local/bin' ডিরেক্টরি থেকে 'mplayer' বাইনারিটা কপি করে নিলাম স্ল্যাকওয়ার সিস্টেমে। এবং স্ল্যাকওয়ার সিস্টেমে বুট করে তাকে চালাতে গেলাম। চলল না, প্রমাদ-বার্তা মানে এরর মেসেজ দিল।

```
mplayer: error while loading shared libraries: libdv.so.2:
cannot open shared object file: No such file or directory
```

অথচ, স্ল্যাকওয়ারে যখন কম্পাইল করে নিলাম, স্ল্যাকওয়ার সিস্টেমে আবার তার নিজের '/usr/local/bin' ডিরেক্টরিতে একই রকম একটা 'mplayer' বলে বাইনারি তৈরি হয়ে গেল। এবং সে একদম স্বাভাবিক ভাবেই চলল।

এখানে আমি আপনাদের দুটো প্রশ্ন করব, যার একটারও উত্তর দেবনা। এক, এই প্রমাদবার্তাটার আগামাথা কোনো আন্দাজ করতে পারছেন কি? আর দুই, স্ল্যাকওয়ার সিস্টেমে যেটা '/usr/local/bin' ডিরেক্টরি সেটা আমার সুজে সিস্টেমে কী ডিরেক্টরি হবে? মনে করুন, আগেই বলেছি, আমার স্ল্যাকওয়ার পার্টিশনটা সুজে সিস্টেমে বুট হয় '/mnt/slackware' ডিরেক্টরিতে। অর্থাৎ, প্রোগ্রামিং ভাষায় সি-তে লেখা 'mplayer' প্যাকেজের সোর্স কোড কম্পাইল করে একটা সিস্টেমে বানানো বাইনারি অন্য সিস্টেমে চলে-না, যেমনটা ঘটার কথা। অথচ ব্যাশ স্ক্রিপ্টগুলো খুব আরামসে চলে এমন যে কোনো সিস্টেমেই যেখানে ব্যাশ আছে। নিজে চালিয়ে দেখুন। ব্যাশ স্ক্রিপ্টিং ভাষায় সচরাচর খুব জাম্বো সাইজের প্রোগ্রাম করা হয়না, তার জন্যে আছে পার্ল (Perl), লিস্প (Lisp), টিসিএল (Tcl) ইত্যাদি।

আমাদের এক নম্বর দিনে আমরা একটা কুচো সি প্রোগ্রাম তুলে দিয়েছিলাম, সেই প্রোগ্রামটা আর একবার তুলে দিই এখানে, তারপর সেই প্রোগ্রামটাই আমরা ব্যাশ দিয়ে করব। এটা একটা চালু প্রথা যে কোনো প্রোগ্রামিং-এর আলোচনার, এই ধরনের একটা কথা-ফুটিয়ে-তোলা প্রোগ্রাম দিয়ে শুরু করা।

```
#include <stdio.h>
int main(void)
{
   printf("\nKire Gadha!!!\n");
   return 0;
}
```

এই কোডটাকে, যেমন এক নম্বর দিনে আমরা বলেছিলাম, আপনি 'gadha.c' নামে সেভ করতে পারেন। নামের শেষে ওই '.c' এক্সটেনশনটা হল সি-প্রোগ্রাম হিশেবে চিনে নেওয়ার চালু প্রথা। এবার এটাকে কম্পাইল করে নিতে পারেন জিসিসি (gcc) দিয়ে। তার জন্যে আলাদা করে কিছু ইনস্টলও করতে হবেনা। জিসিসি কম্পাইলার আপনার গ্নু-লিনাক্স সিস্টেমে গোড়া থেকেই ইনস্টল হয়ে আছে। যে ডিরেক্টরিতে আপনি 'gadha.c' ফাইলটা সেভ করেছেন সেই ডিরেক্টরিতে দাঁড়িয়ে আপনি যদি কমান্ড দেন '`gcc -o gadha gadha.c`', তাহলে জিসিসি ওই ডিরেক্টরিতে 'gadha' নামে একটা বাইনারি ফাইল বানিয়ে দেবে। এই '`-o`' অপশনটা দিয়ে আপনি বাইনারিটা যে 'gadha' নামে বানাতে হবে সেটা জিসিসিকে দিয়ে দিলেন। যদি আপনি না দেন, তাহলে জিসিসি বা গ্নু-কম্পাইলার-কালেকশন বাইনারিটা বানাবে 'a.out' নামে। এবং পরের বার আর একটা কোড ফাইল কম্পাইল করলে তার 'a.out' ফাইলটা সেভ করে দেবে এই 'a.out' ফাইলের জায়গায়। এই নামটা কেন 'a.out' হয় তার আবার একটা গল্প আছে — যাকগে, সে অন্য কথা। এবার এই বাইনারিটাকে চালাতে হবে, এই ডিরেক্টরিতে দাঁড়িয়ে চালাতে গেলে কী কমান্ড দিতে হবে সেটা আমরা জানি, '`./gadha`'। কমান্ড প্রম্পটে এই আদেশটা দিয়ে এন্টার মারলেই স্ক্রিনে লেখা ফুটে উঠবে, উপরে এক লাইন ফাঁকা জায়গা রেখে।

```
Kire Gadha!!!
```

এবার ধরুন এই গোটা কাজটাই আমরা সি নামের প্রোগ্রামিং ল্যাংগুয়েজ দিয়ে না-করে ব্যাশ নামের স্ক্রিপ্টিং ল্যাংগুয়েজ দিয়ে করতে চাইছি। তখন আমাদের একটা ব্যাশ স্ক্রিপ্ট বানাতে হবে। একটা ফাইলে নিচের লাইনদুটো ঢোকালাম আমরা।

```
#!/bin/bash
echo -e "\nKire Gadha"
```

এবং এই ফাইলটাকে সেভ করে দিলাম 'gadha.sh' নামে। এবার পরের স্টেপদুটোও আমাদের পরিচিত। এক নম্বর, এক্সিকিউটেবল করা। নানা ভাবে করা যায়। তার একটা হল '`chmod +x gadha.sh`'। বা, সংখ্যা দিয়ে করতে চাইলে, '`chmod 700 gadha.sh`'। নিজে 'chmod'-এর ম্যানুয়াল পড়ে মিলিয়ে নিন, আগেও আমরা কথা বলেছি এটা নিয়ে, সাত নম্বর দিনে।এবার দু-নম্বর স্টেপ হল এটা চালানো, মানে '`./gadha.sh`'। এবারেও ফলাফল সেই একই। অন্তত লাইনের নিরিখে স্ক্রিপ্টটা বহু ছোট, আর কম্পাইল করার কোনো ঝামেলা নেই। এবং যে কোনো অপারেটিং সিস্টেমে চলবে, সেখানে ব্যাশ থাকলেই। সি-তে কম্পাইল করা 'gadha' বাইনারিটা তা নয়, প্রতিটি আলাদা অপারেটিং সিস্টেমের জন্যে তাকে আলাদা করে কম্পাইল করে নিতে হবে। প্রোগ্রামিং জগতের সমস্ত

সমস্যা যদি এই মেশিনকে 'কিরে গাধা!!!' ডাকার কাছাকাছি সরলতার হত, তাহলে যে ব্যাশ স্ক্রিপ্ট দিয়েই আমাদের যাবতীয় কাজ চলে যেত, অন্য কোনো প্রোগ্রাম করার মাধ্যম তৈরিই হত না, এতে কোনো সন্দেহ নেই। জটিলতার নিরিখে প্রোগ্রামিং ল্যাংগুয়েজরা ব্যাশের থেকে অনেকটা এগিয়ে থাকলেও ব্যাশের নিজের কিছু জোরের জায়গা থাকেই। এক, আলাদা করে চালাতে হয়না। আপনার গ্নু-লিনাক্স চলছে মানে তার উদার উদরের ভিতরে ব্যাশও চলছে, এমনিতেই। এবার, নতুন করে কোনো ব্যাশ স্ক্রিপ্ট চালানো মানে, একটা নতুন ব্যাশ প্রক্রিয়া বা প্রসেস, যা অনেকটা অব্দি মেমরি তথা অন্যান্য রসদ বাঁটোয়ারা করে নেবে, শেয়ার করবে ইতিমধ্যেই বাফারে থাকা ওই প্রারম্ভিক ব্যাশ প্রক্রিয়ার সঙ্গে। কোনো কম্পাইলার, বা রুবি বা পার্ল ইত্যাদি অন্য কোনো ইন্টারপ্রিটার মানেও সেটা তো কিছুটা আলাদা মেমরি নেবেই। ব্যাশের দুই নম্বর প্লাসপয়েন্ট এই যে ব্যাশ আমাদের এমনিতেই চেনা জিনিষ। সিস্টেমের মধ্যে কাজ করা মানেই ব্যাশ শিখতে থাকা। পাঁচ নম্বর দিন থেকে এই অব্দি আমরা যা যা কমান্ড লাগিয়ে সিস্টেমে যা যা করেছি, তা সবই তো ব্যাশ। জ্যান্ত রকমে ইন্টারাকটিভলি ব্যাশকে ব্যবহার করে আসছি, আলাদা আলাদা কেজো কমান্ডের এককে। সে ফাইল কপি করা হোক, নড়ানো হোক, বা পাইপ করা রিডাইরেক্ট করা ইত্যাদি কিছু হোক। এবার, ব্যাশকে যখন ইন্টারাকটিভ রকমে না-ভেবে, একটা স্ক্রিপ্টিং ল্যাংগুয়েজ হিশেবে ভাবছি, এই কমান্ডগুলো এই আকারেই থাকছে, শুধু তাদের একত্রে আনার আলাদা কিছু নিয়ম চালু হচ্ছে মাত্র। তাই ব্যাশ আলাদা করে শেখা শুরু করতে হয়না। ম্যাক্সিমাম কিছু বাড়তি খুঁটিনাটি জেনে নিতে হয়, যখন যা দরকার। তিন, ব্যাশের গোটা ডকুমেন্টেশনটাই ইতিমধ্যেই সদাসর্বদাই রয়েছে আপনার সিস্টেমে। 'man bash' বা 'info bash' করুন এবং পড়ে যান। আর পাশের কনসোলে সেগুলো করে করে দেখতে থাকুন। এক্সউইনডোজে থাকলে, আর একটা টার্মিনাল উইনডোয়।

এই যে ব্যাশ স্ক্রিপ্টটা আমরা এইমাত্র চালালাম, 'gadha.sh' — এটা আসলে কী? নিছক একটা কমান্ড, 'echo' নামের প্রোগ্রামটা ব্যবহার করে, যাকে কমান্ড প্রম্পটে টাইপ করে দিয়ে এন্টার মারলে হুবহু একই কাজ করত। তার মানে আমাদের স্ক্রিপ্টটা আর কিছু করেনি, নিছক একটা সিস্টেমে থাকা কমান্ডকে ব্যবহার করেছে। ব্যাশ কোড ঠিক এটাই করে, পরপর সিস্টেমে থাকা এক বা একাধিক কমান্ডকে তাদের অপশন সহ বিভিন্ন ভ্যারিয়েবলের উপর প্রয়োগ করে যায়। নয় নম্বর দিনে 'writefiles' নামের ব্যাশ স্ক্রিপ্টটা আমরা লাইন ধরে ধরে আলোচনা করেছিলাম। স্ক্রিপ্টের লাইনকটা মনে করুন।

```
#!/bin/bash
echo Name of file?
read f
c=1
d=*****************************************
for i in *
    do echo -e "\n\n$d\n$c. $i\n" >> $f
    cat $i>> $f
    c=`expr $c + 1`
done
```

এই স্ক্রিপ্টটা 'gadha.sh'-এর চেয়ে একটু বড়, ব্যবহার করেছে চারটে কমান্ড 'echo', 'read', 'cat' এবং 'expr' তাদের অপশন সহ। এবং এই কাজগুলো করে চারটে ভ্যারিয়েবলের উপর, লিখবে যে ফাইলে তার নাম মানে 'f', কাউন্টার মানে 'c', বিভাজক বা ডিভাইডর মানে 'd', এবং ডিরেক্টরির ফাইলগুলো একের পর এক মানে 'i'। আর এই সমস্ত আদেশগুলোকে সঠিক বোধ্য রকমে ব্যাশের কাছে পৌঁছে দেওয়ার জন্যে আমাদের লেগেছে কিছু বিশেষ যতি চিহ্ন। যেমন, মন্তব্যবাচক চিহ্ন '#', বা চালান করা মানে রিডাইরেকশন চিহ্ন '>>', বা কমান্ড সাবস্টিটিউশনের চিহ্ন ব্যাককোট '`', বা অ্যাসাইনমেন্ট চিহ্ন '=', বা ভ্যারিয়েবলের মানবাচক চিহ্ন '$'।

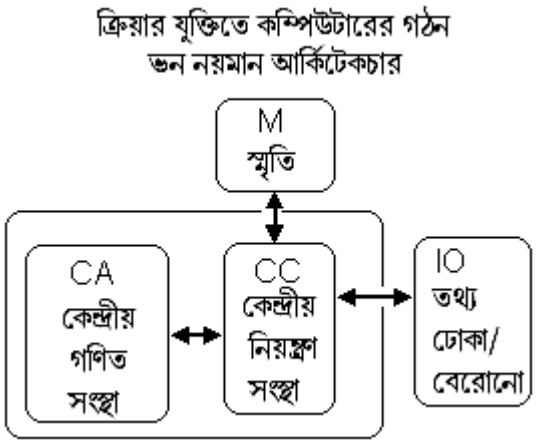
এই গোটাটাকে মিলিয়ে হল ব্যাশ নামের স্ক্রিপ্টিং ল্যাংগুয়েজ। তার কমান্ডগুলো কারা? অন্য প্রোগ্রামিং ভাষায় যেমন কমান্ডগুলোকে আলাদা করে শিখতে হয়, এখানে তা নয়, প্রতিমুহূর্তে একটা গ্নু-লিনাক্স সিস্টেমে কাজ করতে গিয়ে আমরা যে কমান্ডগুলো আদেশগুলো ব্যবহার করে আসছি সেই পাঁচ নম্বর দিন থেকে, সেই কমান্ড গুলোই একদিকে যেমন ব্যাশ নামক কমান্ড প্রম্পটের কমান্ড, অন্যদিকে তেমনি ব্যাশ নামক ল্যাংগুয়েজেরও কমান্ড। আর ভ্যারিয়েবল গুলো নিয়ে এবং দু একটা চিহ্ন নিয়ে আমরা তো আলোচনা একটু আগেই সেরে এলাম। এবার আমরা ব্যাশ শেলের

ভাষাগত এবং ক্রিয়াগত নিয়মকানুনগুলো আর একটু খুঁটিয়ে শিখব। যাকে আমরা বলব কন্ট্রোল স্ট্রাকচার বা নিয়ন্ত্রণ কাঠামো। মনে করে দেখুন, শূন্য নম্বর দিনের সেই আলোচনা, যা আবার আডার কথায় ফেরত এসেছিল, তিন নম্বর দিনে, ক্যালকুলেটর থেকে কী সেই নাটকীয় উল্লম্ফন যা কম্পিউটারকে কম্পিউটার করল? সেটা এই নিয়ন্ত্রণ কাঠামো।

১০।। ভন নয়মান আর্কিটেকচার এবং নিয়ন্ত্রণ কাঠামো

শূন্য নম্বর দিনে আমরা ভন নয়মানের উল্লেখ করিনি ইচ্ছে করেই। স্টিফেন হকিংয়ের ব্রিফ হিস্ট্রি অফ টাইমের ভূমিকাতেই ছিলনা সেই প্রসঙ্গটা, গাণিতিক সমীকরণের সংখ্যার বর্গের ব্যাস্তানুপাতে কমে যেতে থাকে পাঠকের সংখ্যা? খটোমটো এবং অপরিচিত নামের বেলাতেও একই ভাবে সেটা সত্যি। আর সঠিক জায়গায় সঠিক নাম ত্যাগ করে (মলত্যাগের সঙ্গে যদি কোনো আত্মীয়তা পান, সেটা আমার বানানো নয়, ইংরিজিতে বলে নেম-ড্রপিং), সঠিক লোকের কাছে পৌঁছে দেওয়া, লাইনেই আছি দাদা, আত্মবিশ্বাসহীনতার এই কলোনিয়াল সংস্কৃতিটারও বাইরে যেতে চাইছিলাম। সাত নম্বর দিনে খুব উত্তেজিত ভাবে লিখেছিলাম না, নেম-ড্রপিং সিম্বল-ড্রপিং করে আবহটাকে ইচ্ছে করে আরো খটোমটো আরো এলিট করে দেওয়া এটাও একটা রাজনীতি। আমাদের ঘরের কোনো টুম্পা যাতে আডা লাভলেস হওয়ার স্পর্ধা করার স্বপ্নেও পৌঁছতে না-পারে তার রাজনীতি। যাতে লিনাক্স এইট করা লিনাক্স বিশারদ ধেড়ে পাঠারা সাহেব অ্যাকাডেমির সাহেব ড্রপিং ঘেঁটে যেতে পারে, আর তার সৌরভে আমোদিত হতে পারে বন্দুকের আওয়াজে বমকে যাওয়া স্যাভেজের বাচ্চারা — দশদিক থেকে যারা আসিতেছে চলে পিসি পিসি শিখবে বলে। মার্জিন অফ মার্জিন বলে আমাদের একটা বইয়ে একটা শব্দ আছে 'savage', যাকে বানানে লেখা 'sa**V**Age'। প্রথম আর শেষটুকু মিলিয়ে হয় সেজ, সাধক, তপস্বী, প্রাজ্ঞ। আমি ব্যক্তিগতভাবে একজন হতভাগ্য পরিণামহীন ভবিষ্যতহীন মানুষ, একদম জৈবনিক অর্থেই। আজকাল মনে হয়, আমাদের সমস্ত সমাজ বদলানোর স্বপ্নই আত্মহত হয়েছে চালু ইতিহাসের হাতে, নিজেকে নিজের জীবনকে নিয়ে তাই আর কিছু এসে যায়না, নাউ দ্যাট আই অ্যাম নাইন্টিথ্রি, আই কেয়ার এ ড্যাম ইউ সি, কিন্তু একটা টুম্পাও যদি আডা হতে চেষ্টা করার উত্তেজনাটাও পেতে পারে — তার চেয়ে বড় পাওনা আর কী হয়? একটা স্যাভেজও যদি এই নিজের মধ্যে সঙ্গোপন সেজ খোঁজার প্রক্রিয়ায় পৌঁছতে পারে, তার একটুও রসদ পায় আমার কাজ থেকে। এই গোটা বইটায় প্রতিমুহূর্তে আপনাকে এগিয়ে দেওয়ার চেষ্টা করা আমাদের অ্যাকাডেমির ধেড়ে পাঠাদের সেই মহিমার রাজনীতির একটা বিপরীত রাজনৈতিক চেষ্টার একটা ব্যক্তিগত যাত্রা। সেই যাত্রায় সঙ্গীও তো জুটেছে এই গ্নু-লিনাক্স জগতের থেকেই, অন্তত পাঁচটা লোক সত্যিই উত্তেজিত এই বইটা ঘিরে — সঙ্কর্ষণ, অশেষ, তথাগত, পিউ আর দেবাশিস। আরো অনেক বেশি জীবনীশক্তি নিয়ে আরো অনেক জীবন্ত রকমে যখন সমাজ বদলানোর স্বপ্ন দেখেছিলাম, তখন তো এটুকু উত্তেজনাও তৈরি করতে পারিনি নিজের চারপাশের রাজনীতিতে।

ফেরত আসি সেই নাম না-করা ভন নয়মান আর্কিটেকচারের কথায়। আমরা কাজের ভিত্তিতে একটা কম্পিউটারের পাঁচ রকমের উপাদানের কথা বলেছিলাম। এক, তথ্য ঢোকানো বা ইনপুট ডিভাইস। দুই, তথ্য বার করা বা আউটপুট ডিভাইস। তিন, তথ্য চটকানো বা প্রসেসিং ডিভাইস। চার, তথ্য ধারণ বা হোল্ডিং ডিভাইস। আর পাঁচ, সবচেয়ে গুরুত্বপূর্ণ, এক থেকে চার এই চারটে কাজের নিয়ন্ত্রণ বা কন্ট্রোল ডিভাইস। এবার এই ছবির সঙ্গে মিলিয়ে ভন নয়মানের মূল ছকটা দেখুন। শুধু একটা জিনিষ, এখানে ইনপুট আর আউটপুটের উপাদানগুলোকে তাদের কাজের ধরনের আত্মীয়তার ভিত্তিতে একটা উপাদান হিশেবে ধরা হয়েছে।



কম্পিউটার বিজ্ঞানে ভন নয়মানের অবদানের কথা আমরা আগেই উল্লেখ করেছি, চার নম্বর দিনে। এখানে আমরা শুধু পেনসিলভানিয়া বিশ্ববিদ্যালয়ে থাকাকালীন এডভ্যাক (EDVAC) কম্পিউটারের সূত্রে ১৯৪৫ সাল নাগাদ তিনি যে ছকটা দিয়েছিলেন স্টোরড-প্রোগ্রাম-কম্পিউটারের লজিকাল গঠনের, সেটাকেই তুলে দিচ্ছি। স্টোরড-প্রোগ্রাম বিষয়টা খেয়াল করুন, এর মানে যেখানে গোটা প্রোগ্রাম বা আদেশ-

তথ্যটাকেই স্মৃতিতে তুলে কম্পিউটার কাজ করতে পারে। কম্পিউটারকে এখানে মূল চারটে অংশে ভাগ করা হয়েছে — সেন্ট্রাল অ্যারিথমেটিকাল ইউনিট বা কেন্দ্রীয় গণিত সংস্থা (CA), সেন্ট্রাল কন্ট্রোল ইউনিট বা কেন্দ্রীয় নিয়ন্ত্রণ সংস্থা (CC), মেমরি বা স্মৃতি (M) এবং ইনপুট/আউটপুট বা তথ্য ঢোকা/বেরোনোর সংস্থা (IO)। 'CA' করবে গাণিতিক হিশেবের কাজ। চারটে মূল পাটীগাণিতিক ক্রিয়া, মানে যোগ বিয়োগ গুণ ভাগ, এবং তাদেরকে একত্রে কাজে লাগিয়ে সূচক বা মূলের, মানে ইনডেক্স বা রুটের, তথা লগ এবং ত্রিকোনমিতিক ফাংশনের বা অপেক্ষকের হিশেবের কাজ। 'M' ধরে রাখবে তথ্য, গাণিতিক তথ্য মানে বিভিন্ন সংখ্যা, যাদের নিয়ে সে কাজ করবে, এবং কী ভাবে কী কাজ সে করবে তার নির্দেশের গোটা তথ্যটাও তোলা থাকবে এই 'M'-এ। এই নির্দেশ তথ্যকে এখন আমরা চিনি — বাইনারির ভিতরের কম্পাইলড কোড বা ইন্টারপ্রিট করা স্ক্রিপ্টেড কোড। আর 'IO' মানে হল ব্যবহারকারীর সঙ্গে কম্পিউটারের পারস্পরিক সম্পর্কের ইন্টারফেস, কাঁচামাল তথ্য ঢোকানো এবং উৎপাদিত তথ্য বার করার। আর 'CC' করবে অন্য কাজ গুলোকে নিয়ন্ত্রণের কাজ। কোন তথ্যের উপর কোন নির্দেশ কাজ করবে, কতদূর অব্দি কী কাজ শুরু করবে, কখন কোন কাজ শুরু বা শেষ হবে, কখন কোন কাজ বদলাবে, ইত্যাদি।

ভন নয়মানের এই ছকটা একটা লজিকাল বা যুক্তিবৈজ্ঞানিক ছক। মেশিন বা যন্ত্রাংশের ভৌত গঠন নিয়ে মাথা ঘামাননি নয়মান। কম্পিউটার ক্রিয়ার যুক্তিগত প্রবাহটাকে ধরতে চেয়েছিলেন। এবার একটা অন্যরকম ক্রিয়াকে ভাবুন। ধরুন আপনি কিছু হিশেব কষছেন ক্যালকুলেটর নিয়ে। তখন যে কাগজে রাফ হিশেবগুলো লিখছেন সেটা নিশ্চিতভাবেই 'M'। ক্যালকুলেটরের বোতামগুলো আর ডিসপ্লেটা মিলিয়ে তার 'IO'। ক্যালকুলেটর তথা তার অভ্যন্তরে সার্কিটটা হল 'CA'। কিন্তু 'CC' কই? 'CC' কী নয়, প্রশ্নটা এখানে হবে, 'CC' কে? কর্ম নয়, কর্তা। এখানে আপনি হলেন 'CC'। সচেতন নিয়ন্ত্রক সংস্থা। ক্যালকুলেটরটা ক্যালকুলেটর বলেই একটা সচেতন নিয়ন্ত্রক সংস্থার প্রয়োজন পড়ছে। আর স্টোরড-প্রোগ্রাম-কম্পিউটারের আপনি সেই নিয়ন্ত্রণটাকে বকলমা দিয়ে দিচ্ছেন, পাওয়ার-অফ-অ্যাটর্নি করে দিচ্ছেন কম্পিউটারটাকে। প্রোগ্রামিং ল্যাংগুয়েজ-এর বেলায় কোড এবং কম্পাইলার এবং বাইনারির যৌথ হাতে। আর ব্যাশ স্ক্রিপ্টিং-এর বেলায় ব্যাশ স্ক্রিপ্ট এবং ব্যাশের যৌথ হাতে। এরাই হয়ে দাঁড়াচ্ছে ওই সচেতন নিয়ন্ত্রণের প্রতিনিধি। আর নিয়ন্ত্রণের ভৌত ক্রিয়াটা ঠিক কার হাতে থাকছে সেটা যদি আপনি মাথায় আনতে না-পারেন, তাহলে বড্ড ভুল জায়গায় পড়ছেন, আপনার ফের শূন্য থেকে শুরু করার কথা।

এবং এবার আমাদের ব্যাশ নামক স্ক্রিপ্টিং ভাষাটার ব্যাকরণটা একটু জানতে হবে, জ্যান্ত মানুষের সচেতন নিয়ন্ত্রণটাকে সে ঠিক কী ভাবে বাস্তবে প্রয়োগ করে সেটা জানতে।

১১।। চিহ্নপ্রবাহ এবং তার দিকনির্দেশ

মাউন্ট ব্যাটন সাহেবঅ, তোমার সাধের ব্যাটন কার হাতে থুইয়া গেলাঅ — আজাদিপ্রাপ্তির পর চারদিকের ব্যাপার স্যাপার দেখার বেজায় বেদনা ছিল হেমাঙ্গ বিশ্বাসের গানে, আমাদের এরকম আক্ষেপের কোনো কারণ নেই। আমাদের ব্যাটন আমরা সাভিলাষ সারাম দিয়ে দিয়েছি ব্যাশকরপুটে। ব্যাশ এবার গোটাটা দেখভাল করছে, মেশিনকে ঘিরে গোটা ক্রিয়াকলাপের। এই ক্রিয়াকলাপটা আসলে কী? সেই শূন্য এক দুই তিন করে যে দিনগুলো আমরা পেরিয়ে এলাম, তার গোটাটা মাথায় রেখে একবার ভাবুন তো? শূন্যয় সেই বলেছিলাম না, শেষ অব্দি একটা কম্পিউটারের ক্রিয়াকলাপ খুব সরল সাধাসিধে, ডিরেক্ট। কোনো ধূসরতার সন্ধ্যাভাষার ধুন্ধলে শাহি উপাখ্যান সেখানে ছুপা নেই। গোটাটাই বাইনারি, শূন্য আর এক — এই হল কম্পিউটারের তথ্য মানে ডেটা। এর মাঝে আমার মেশিনের হার্ডডিস্কটা ক্র্যাশ করল, দিন পাঁচ ছয় সাতে যে দুটো চল্লিশ জিবি করে হার্ডডিস্ক '/dev/hda' আর '/dev/hdb'-র কথা বলেছি, এর প্রথমটা গেল। একটা নতুন হার্ডডিস্ক কিনে আনলাম টাকা ধার করে, তবে দাম বেজায় কমে গেছে এখন, আশি জিবি ৭২০০ আরপিএম, তারই দাম নিল চার হাজার তিনশো। আরপিএম বিষয়টা মনে করতে পারছেন? সাত নম্বর দিনে ছিল। তা যাই হোক, নিজেই লাগালাম, বেজায় আমোদ পেয়েছি তাতে, নিজেকে বেশ কাবেল লাগছিল। দেখবেন, যে কাজটা আপনার এলাকার বাইরে, সেটা করতে পারলে ভারি ভালো লাগে। তা, এই একশোকুড়ি জিবি জোড়া মোট ডিস্কভূমির মোট ব্যবহৃত এই মুহূর্তে ৩২.৫ জিবি। এটা পেলাম সবগুলো পার্টিশনে মাউন্ট করার পর '`df -h`' মেরে, প্রতিটি পার্টিশনের ব্যবহৃত ভূমিগুলো যোগ করে। ৩২.৫ গুণ ১০২৪ এমবি, গুণ ১০২৪ কেবি, গুণ ১০২৪ বাইট, গুণ ৮ বিট। মোট দাঁড়াল, ১২০ জিবির ভিতর ৩২.৫ জিবি — ১.০৩০৮ × $১০^{১২}$ বিটের ভিতর ২.৭৯১৭ × $১০^{১১}$ বিট

মানে, মোটামুটি ধরুন ২৮-এর পরে দশটা শূন্য দিয়ে যত বড় সংখ্যা ততগুলো ছোট ছোট খোপ, যাতে হয় ১ আছে নয় ০ — এটা হল আমার মেশিনে এই মুহূর্তে বসবাসমান মোট তথ্যের পরিমাণ। আটাশ হাজার কোটি কুচো কুচো চৌম্বক খোপ যার প্রত্যেকটায় হয় ১ থাকতে পারে নয় ০। খোপের সংখ্যাটাকে বাড়িয়ে তোলা যাবে, আর হার্ডডিস্ক না বাড়িয়ে, এক লক্ষ কোটি অব্দি। একটু চ্যাংড়ামি করে বলা যায়, আমার মেশিনের মোট হার্ডডিস্কভূমিতে মোট যত শূন্য বা এক লেখা যায়, সেটা যদি আমায় হাতে করে লিখতে হত, আর মিনিটে যদি আমি একশোটা শূন্য বা এক লিখতে পারি, এই পোস্টচল্লিশে আর কত স্পিডে লিখব, আমার মাত্র ১৯০২৬ বছর লাগত। মানে এক বছরের মধ্যে গোটাটা লিখতে হলে, নিজে লেখার পর, আমার মাত্র উনিশ হাজার পঁচিশ জন সহকারী লাগত। তিন নম্বর দিনে আমরা লিখেছিলাম মনে আছে, অ্যাবাকাসের উদ্ভবের কথায়, নিজে এইসব করতে করতে মানুষের মনে হল, একটু সহকারী রাখার। সেই একটু সহকারী মানে মাত্তর উনিশ হাজার পঁচিশ জন। তাও এটা শুধু লেখার জন্যে। ব্যারন দ প্রনির হিউম্যান কম্পিউটারদের মোট কতজনের হিশেবের কাজ করে এক একটা মেশিন, সেই হিশেবটা নয় কিন্তু এটা।

কাজের কথায় আসা যাক। এই আটাশ হাজার কোটি খোপ জুড়ে কী আছে? আছে তথ্য। দুরকম তথ্য। এক, যে তথ্যকে চটকাতে হবে, ধরুন তাকে বলছি কাঁচা বা র তথ্য। আর, আগের সেকশনে যেমন বললাম, কী ভাবে এদের চটকাতে হবে, সেই আদেশ মালা, তারাও তো তথ্য, কম্পিউটারের নাড়িভুঁড়িতে তারাও তো ভরা আছে এই শূন্য আর এক দিয়েই। কারণ কম্পিউটারটা বেজায় গাধা, এই শূন্য বা একের, ভগবান আর শয়তানের, বাম আর রামের বাইনারি ছাড়া আর কিছু ব্যাটা বোঝেই না, মধ্যের ধূসরকে বুঝবে কী করে, আমাদের মত পোস্টমডার্নিজম পড়েনি তো। এই নির্দেশ তথ্য, ইন্সট্রাকশন ডেটা, সেটাও তো তথ্য, যাকে নয়মান স্টোরড প্রোগ্রাম বলে ডেকেছিলেন। এই প্রোগ্রাম মানে আমরা জানি, নানা জাতের নানা কিসিমের বাইনারি। কেউ শুধু সাঙ্গীতিক তথ্য চটকান মানে অডিও প্যাকেজ, কেউ শুধু লিখিত শব্দ চটকান মানে ওয়ার্ড প্রসেসর, কেউ শুধু মেশিন থেকে মেশিনে ভ্রাম্যমান তথ্য চটকান মানে ব্রাউজার, ইত্যাদি।

তাহলে কম্পিউটারের মোট ইতিহাস এবং ভূগোল কী দাঁড়াল? তার পেটের মধ্যে ভরা কোটি কোটি অর্বুদ অর্বুদ শূন্য বা এক লেখার খোপ — এটা তার ভাণ্ডার। এবং তার কাজ মানে কী? এই কোটি কোটি খোপে ভরা আরব আরব শূন্য বা এক ইধার-কা-মাল-উধার করা। ধরুন গাণিতিক একটা প্রতিতুলনা ব্যবহার করা যাক। এই অগণ্য শূন্য বা এক, এরা থিতু হয়ে আছে, এর মানে তথ্য। তথ্য তখন স্ক্যালার। আর বৌবাজার জেলেপাড়ার বাজারের শানে 'কাঁটাপোনা লরচে', মালেরা নড়ছে — এক ঠাঁই থেকে আর এক ঠাঁই যাচ্ছে, এর মানে এদের চটকানো হচ্ছে, মানে আমাদের মেশিন চলছে, কাজ করছে, প্রোগ্রামরা রান করছে। তথ্য তখন ভেক্টর। তার একটা গতি আছে দিকনির্দেশ আছে। তুলনাটা খুব একটা জমল কি? বুঝতে পারছিনা, সঙ্কর্ষণদের কাছ থেকে গালাগাল খাওয়া বা না-খাওয়াতেই বুঝতে পারব। আর কোনোদিন আমি এদের মত ছোটলোকদের সঙ্গে মিলে কিছু লিখছি না। আজ ভোর সোয়া পাঁচটায় আমায় ফোন করে বলেছে, এই সব খাজা মাল ছড়াচ্ছ কেন? দেড় দশকের বড় একটা লোকের সঙ্গে কথা বলার কী সহবত।

এবার দিন নম্বর একের সেই ওল্টানো সৌধের মত হার্ডওয়ার থেকে সফটওয়ারের হায়েরার্কির ছকটা আর একবার তুলে দিই এখানে। মনে করুন, একদম নিচের স্তরে, হার্ডওয়ারের স্তরে ভৌত চৌম্বকতার শূন্য আর এক, তাদের সঙ্গে অন্য ভৌত উপাদান, এর উপরে মাইক্রোআর্কিটেকচারের স্তরের ভিতটা, যেখানে দাঁড়িয়ে আছে মেশিন ল্যাংগুয়েজ, এখানেই হার্ডওয়ার বা যন্ত্রপাতির এলাকা শেষ। এবার মূল কাঠামো বা

<table>
<tr><td>(৬) হিসাবনিকাশের সফটওয়ার</td><td>(৬) রেলের টিকিট বিলিব্যবস্থা</td><td>(৬) ইন্টারনেটে ঘোরাঘুরি</td><td rowspan="2">প্রায়োগিক সফটওয়ারগুলো বা অ্যাপ্লিকেশনস</td></tr>
<tr><td>(৫) কম্পাইলার</td><td>(৫) এডিটর</td><td>(৫) কমান্ড ইন্টারপ্রিটার</td></tr>
<tr><td colspan="3">(৪) অপারেটিং সিস্টেম</td><td rowspan="4">মূল কাঠামো বা সিস্টেম প্রোগ্রামের এলাকা
হার্ডওয়ার বা যন্ত্রপাতির এলাকা</td></tr>
<tr><td colspan="3">(৩) মেশিন ল্যাংগুয়েজ</td></tr>
<tr><td colspan="3">(২) মাইক্রোআর্কিটেকচার</td></tr>
<tr><td colspan="3">(১) ভৌত উপাদানগুলো</td></tr>
</table>

সিস্টেম প্রোগ্রামের এলাকা শুরু। এর মধ্যে আবার দুটো স্তর, নিচের ভিতটা হল অপারেটিং সিস্টেম, আমাদের গ্নু-লিনাক্স কারনেল। তার উপরে তিনটে সমান্তরাল জিনিষ — কম্পাইলার, এডিটর, কমান্ড ইন্টারপ্রিটার। আমরা এখন এদের চিনি, কম্পাইলার মানে আমাদের জিসিসি, এডিটর মানে ইম্যাক্স ভিম গোছের কেউ, আর কমান্ড ইন্টারপ্রিটার, হেইডা কেডা বলেন তো? এই দিনের আলোচনায় খাড়াইয়া আপনি যদি না-পারেন, তাইলে ব্যাস, খ্যামা দ্যান, মেশিনে গান শোনেন, তার বেশি কিছু আপনারে দিয়া হইব এমনডা লাগে না। আর এখানেই শেষ দ্বিতীয় স্তর। তার উপরের স্তরে আছে প্রায়োগিক সফটওয়ারগুলো, প্রতি মুহূর্তে যাদের নিয়ে আমরা কাজ করছি। নানা কাজের নানা বাইনারি বা বাইনারি-সমাহার।

তাহলে দেখুন, ভৌত চৌম্বকতার ওই শূন্য বা একের আকারেই তথ্যকে দেখাটা ফুরিয়ে যাচ্ছে হার্ডওয়ারের স্তরেই, তার পরে থাকছে, সরাসরি আমরা যেভাবে কাজ করি, কিবোর্ডে টাইপ করে যে বর্ণমালা পাই, তাদের সমাহার, এবং নিউলাইন, ক্যারেজ রিটার্ন, বেল ইত্যাদি আরো কিছু চিহ্ন। এদের মিলিয়ে যেটা তৈরি হচ্ছে সেটা আমাদের কাছে বোধ্য, হাইলেভেল হিউম্যান আন্ডারস্ট্যান্ডেবল ভাষায়। যেমন এডিটর কম্পাইলার এবং ব্যাশ ইত্যাদি কমান্ড ইন্টারপ্রিটারে আমরা যেভাবে কাজ করি। আমাদের পড়ার এবং বোঝার মত ভাষায় বলেই আমরা তাদের নিয়ে কাজ করতে পারি। শূন্য নম্বর দিনের তথ্যের পরিমাপ এবং অ্যাসকি সংক্রান্ত আলোচনাটা মনে করুন। কিন্তু এই কম্পাইলার, এডিটর বা ব্যাশ তথা যে কোনো কমান্ড ইন্টারপ্রিটারের মূল বাইনারি বা এদের উপরের স্তরে চালনীয় প্রোগ্রামগুলোর বাইনারি খুলে দেখুন, সেটা আমাদের সম্পূর্ণ অবোধ্য। যেমন, শূন্য নম্বর দিনেই আমরা একটু ভিশুয়াল চ্যাংড়ামি করেছিলাম, ছবি মবি দিয়ে ভিত্তার করে দেওয়া আমাদের ওই বাহারি পিডিএফ, সেটাকে সরাসরি এডিটর দিয়ে খুলে আমরা পাচ্ছিলাম ক্রিপ্টন গ্রহের সংখ্যাগুরু জনগোষ্ঠীর মাতৃভাষায় লেখা কাব্যগ্রন্থ, অথচ সেই জিনিষই পাঠ করে আমাদের সামনে পিডিএফ খুলে দিচ্ছে এক্সপিডিএফ বা গোস্টভিউ ইত্যাদি। কিন্তু আমাদের কাছে বোধ্য হোক, বা অবোধ্য, এই সমস্ত তথ্যই হল চিহ্ন দিয়ে তৈরি, চিহ্ন সমাহার। এই চিহ্ন সমাহার একটা প্রবাহের মত। যখন সেটা স্থির হয়ে আছে তখন সেটা তথ্য, কাঁচা তথ্য বা নির্দেশ তথ্য, র ডেটা বা প্রোগ্রাম। আর যখন মেশিন চলছে, প্রোগ্রামগুলো রান করছে, তখন সেই চিহ্নপ্রবাহ নড়ছে, একটা নির্দিষ্ট গতিমুখে নির্দিষ্ট নিয়ম মেনে, যে নিয়মগুলো ওই প্রোগ্রামে লেখা আছে।

তার মানে কী দাঁড়াল? কম্পিউটারে শেষ অব্দি কী আছে? চিহ্নের প্রবাহ। ক্যারেকটার স্ট্রিম। আর তাই যদি হয়, তাহলে, ভন নয়মানের ছকে আমাদের 'cc' মানে সচেতন নিয়ন্ত্রণের ব্যাটন হাতে নেওয়া মাত্র, ব্যাশকে স্থির করে নিতে হবে, তার চিহ্ন প্রবাহকে সঠিকভাবে প্রবাহিত করার মূল খাতগুলো কী হবে। মূল চলাচলপথের দিকগুলো প্রথম থেকেই ঠিক করে না-দিলে, পরে, প্রোগ্রামগুলো চলাকালীন, চারিদিকে প্রলয়নৃত্য পরায়ন নটরাজের মুদ্রার মত বহুমুখী বিচিত্রমুখী নিরবচ্ছিন্ন চিহ্নপ্রবাহের ট্র্যাফিক সে আর সামলাতে পারেনা। ঠিক এটাই ব্যাশ করে নেয় তার তিনটে প্রাথমিক ফাইল স্ট্যান্ডার্ড ইন, স্ট্যান্ডার্ড আউট আর স্ট্যান্ডার্ড এরর দিয়ে।

স্ট্যান্ডার্ড ইনপুট, স্ট্যান্ডার্ড আউটপুট এবং স্ট্যান্ডার্ড এরর। তথ্যের দিকনিয়ন্ত্রণের মূল উপাদান এই তিনটে। শুধু তথ্যের নয়, আপনার এবং আপনার দাসানুদাস ব্যাশেরও ক্রিয়ার দিকনির্দেশের তিনটে গুরুত্বপূর্ণ ভেক্টর এই স্ট্যান্ডার্ড ইন, সচরাচর ছোট আকারে 'Std. In', স্ট্যান্ডার্ড আউট, 'Std. Out', এবং স্ট্যান্ডার্ড এরর, 'Std. Err.'। সেই পাঁচ নম্বর দিন থেকে আমরা যে বারবার উল্লেখ করছি, আলোচনা করছি, এমনকি না-জেনেই ব্যবহারও করছি, রিডাইরেকশন বা চালান করা এবং খাত বদলানো বা পাইপ করার কথা, সেটা এবারে আমরা ঠিক করে বুঝতে পারব।

আগেই অনেকবার বলেছি, গ্নু-লিনাক্সে সবই এক একটা ফাইল। এই স্ট্যান্ডার্ড ইন আউট এবং এরর — এরাও তিন পিস ফাইল। তিনটে ক্যারেকটার স্ট্রিম — চিহ্নের প্রবাহ। শূন্য নম্বর দিনেই বলেছিলাম, ক্যারেকটার মানে শুধু অক্ষর না, কমা স্পেস ইত্যাদি যতিচিহ্নগুলো, এবং নিউলাইন ক্যারেজ রিটার্ন বেল ইত্যাদি অছাপনীয় চিহ্নগুলোও। তার মানে, আরো ভেঙে বললে বাইটের স্রোত, তিনটের জন্যে তিন রকম স্রোতের তিনটে ফাইল। যে ফাইলগুলোকে ব্যাশ তথা ব্যাশাধীন বিভিন্ন কমান্ড নিজের ইনপুট বা আউটপুট হিশেবে ব্যবহার করে।

সঠিক টেকনিকাল অর্থে বলতে গেলে, কোনো কমান্ডই কিন্তু আপনার কিবোর্ডের কাছ থেকে আদেশ নেয়না, নেয় স্ট্যান্ডার্ড ইন থেকে। ডিফল্ট সেটিং-এ এই স্ট্যান্ডার্ড ইন হল আপনার কিবোর্ড। বা, ওই একই টেকনিকাল অর্থে, কোনো আদেশ বা কমান্ড তার নিজের কাজের ফলাফল আপনার স্ক্রিনে পাঠায় না, পাঠায় স্ট্যান্ডার্ড আউটে, যা

ডিফল্ট সেটিং-এ আপনার স্ক্রিন। আর, কমান্ড বা আদেশ পালন করতে গিয়ে যা যা ত্রুটি ঘটে, তাদেরকে পাবলিকলি জানান দেয় ব্যাশ তার প্রমাদ-বার্তা বা এরর মেসেজ দিয়ে, যার নাম স্ট্যান্ডার্ড এরর। এখুনি এদের ব্যবহারের ব্যাকরণে ঢুকব আমরা, তার আগে এদের কাজকর্মের সঙ্গে একটু পরিচিততর হয়ে নিই, আসুন। আমরা নয় নম্বর দিনের এবং আজকের আলোচনায় বারবার 'cat' করে '.bashrc' ফাইলকে অন্য একটা একটা ফাইলে নিয়েছি। তার জন্যে কমান্ড দিতে হয় '`cat ~/.bashrc>bashrc.text`'। বা '`.profile`' ফাইলকেও করা যায় এরকম। আমরা করেছি বারবার। এবার তার বদলে কমান্ড দেওয়া যাক

```
cat - ~/.bashrc - ~/.profile>local.conf.text
```

এবার এন্টার মারুন। দেখুন, ঠিক '`cat`' কমান্ডটা দেওয়ার পরে যেরকম একটা প্রম্পটহীন কালো স্ক্রিন আসে, সেরকম এসেছে। এই যে '-' চিহ্নটা রাখা হয়েছে দুবার, এটা আসলে দুটো হেডিং বানানোর জন্যে। 'local.conf.text' ফাইলটায় আমরা দুটো ফাইল রাখছি, তাদের দুটো হেডিং দেব এবার, যাতে একটা থেকে অন্যটাকে আলাদা করা যায়। '-' চিহ্নটা ঠিক এটাই করবে। আপনি কালো স্ক্রিনে টাইপ করে যা যা দেবেন, সেই তথ্য এবং ফাইলগুলো পরপর রাখতে রাখতে যাবে 'local.conf.text' ফাইলে। প্রথমে টাইপ করে দিন '`file: '~/.bashrc'`', তারপরে একটা আস্ত লাইন পরপর '=' চিহ্ন টাইপ করে দিন, এবার পরপর দুবার এন্টার মারুন, যাতে একটা লাইন ফাঁকা রাখে '.bashrc' ফাইলের আগে, এবার '<Ctrl><D>' মারুন। দেখুন, স্বাভাবিক 'cat' করার সময় এতেই যেমন কমান্ড প্রম্পট ফেরত আসে, মানে, ক্যাট করার কাজ শেষ হল, এখানে তা হয়নি, এখনো সেই কালো স্ক্রিন। এবার আসলে দ্বিতীয় '-' চিহ্নটার আদেশ পালন করছে ব্যাশ। এবার একই রকম একটা হেডিং টাইপ করে দিন '.profile' ফাইলের জন্যে। এবার '<Ctrl><D>' মারলেই দেখবেন কমান্ড প্রম্পট ফেরত এলো। মানে গোটা কাজটা ফিনিশ। এবার বোঝার চেষ্টা করা যাক। প্রথমে 'local.conf.text' ফাইলে ব্যাশ ক্যাট করল '-' চিহ্ন অনুযায়ী স্ট্যান্ডার্ড ইনপুটকে, যার ডিফল্ট হল কিবোর্ড, মানে কিবোর্ড থেকে আপনি যা যা টাইপ করলেন। তারপরে একটা ফাইল, আবার স্ট্যান্ডার্ড ইনপুট, আবার একটা ফাইল। অর্থাৎ, যা আমরা বলছিলাম, কোনো কমান্ড তার ইনপুট নানা জায়গা থেকে নিতে পারে, স্ট্যান্ডার্ড ইনপুট বা ডিস্ক ফাইল। এমন কি, নিতে পারে অন্য কমান্ডের স্ট্যান্ডার্ড আউটপুট থেকেও।

একটা কমান্ডকে অন্য কমান্ডের আউটপুট থেকে নিজের ইনপুট নিতে আমরা দেখেছি। ছয় নম্বর দিনের ১ নম্বর সেকশনে আমরা যখন '`man -k CD | less`' করে 'CD' শব্দটা কোন কোন ম্যানুয়াল পেজে আছে সেই তালিকাটা পড়ছিলাম, বা '`man -k CD | grep 'audio'`' করে যখন সেই তালিকাটার কোন কোন লাইনে 'audio' শব্দটা আছে সেটা খুঁজে বার করছিলাম, তখন কী ঘটছিল সেটা বোঝার চেষ্টা করুন। 'less' নামের পেজারটা একটা কমান্ড, পাতার এককে একটা চিহ্নপ্রবাহ বা ক্যারেকটার স্ট্রিমকে দেখানোর। সেই '`less`' কমান্ডটা আমরা এখানে তার ইনপুট নিচ্ছে স্ট্যান্ডার্ড আউটপুট থেকে, মানে স্ক্রিন থেকে। স্ক্রিনে যা রেকারিং ডেসিমালের মত ধ্যার‍্যার‍্যার‍্যার‍্যা করে বেরিয়ে যেত, তাকে ভদ্রলোকের পাতে দেওয়ার মত করে বাটিতে বাটিতে সার্ভ করছে 'less'। আর, পরেরটায় আমরা ব্যবহার করছি 'grep', খোঁজার কমান্ড, খুঁজে খুঁজে সেই লাইনগুলোকে বার করে 'grep', যেখানে উদ্দীষ্ট শব্দটা আছে, এখানে সেটা 'audio'। কারণ আমরা শুধু অডিয়ো সিডির ব্যাপারস্যাপারই খুঁজছিলাম। এখানেও দেখুন, স্ক্রিন থেকে মানে স্ট্যান্ডার্ড আউট থেকে তার ইনপুট নিচ্ছে 'grep'। এবং 'less' বা 'grep' দুজনেই তাদের নিজের নিজের ইনপুটের উপর নিজের কাজকর্ম করার পর তাকে পৌঁছে দিচ্ছে কোথায়? স্ক্রিনে মানে স্ট্যান্ডার্ড আউটের ডিফল্টে। আবার এরা এদের নিজেদের আউটপুটকে স্ট্যান্ডার্ড আউটে না-দিয়ে একটা ডিস্ক ফাইলেও দিতে পারত, ওখানেই সেটা করেছিলাম আমরা। পরে ধীরেসুস্থে পড়ার জন্যে একটা টেক্সটফাইল বানিয়ে নিয়েছিলাম।

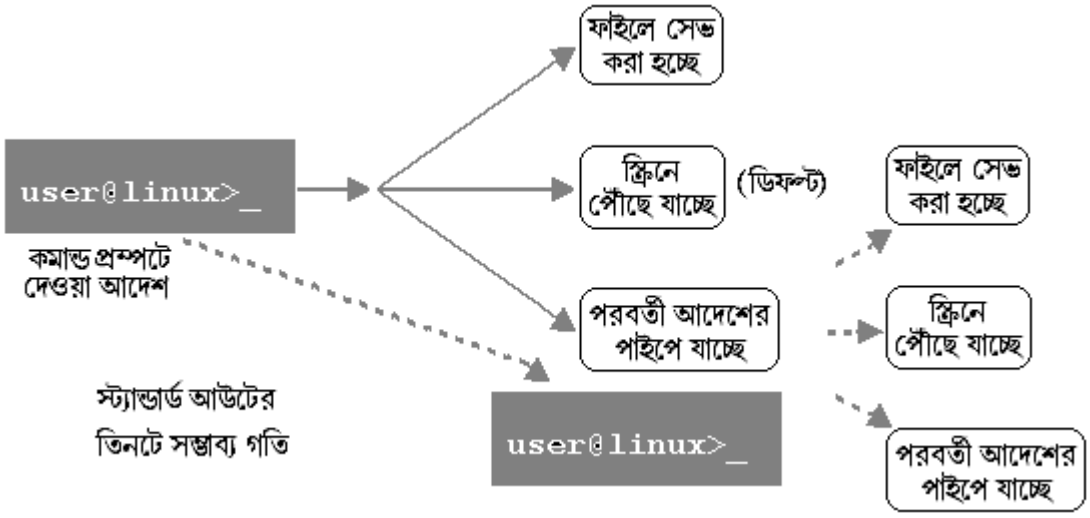
```
man -k CD | grep 'audio'> audio.cd.program.text
```

অর্থাৎ, একটা কমান্ডের কাছে কিছু এসে যায়না, তার ইনপুট কোথা থেকে আসছে — স্ট্যান্ডার্ড ইন না কোনো ডিস্ক ফাইল। বা কোথায় তার আউটপুট পৌঁছে দেবে — স্ট্যান্ডার্ড আউট না কোনো ডিস্ক ফাইল। স্ট্যান্ডার্ড ইন, স্ট্যান্ডার্ড আউট, স্ট্যান্ডার্ড এরর — এরা প্রত্যেকেই এক একটা ফাইল, ক্যারেকটার স্ট্রিম বা চিহ্নের প্রবাহ। স্ট্যান্ডার্ড ইন, স্ট্যান্ডার্ড আউট আর স্ট্যান্ডার্ড এরর — এদের তিনজনের ডিফল্ট তো বলেছি আগেই, প্রথমটার উৎস কিবোর্ড, আর

পরের দুটোর মোক্ষ বা চরিতার্থতা হল কনসোল বা স্ক্রিন বা টার্মিনাল। পরের দুটো যে একই ক্ষুরে মাথা মোড়াবে সেটা এভাবেই ব্যবস্থা করা। এই মাত্র আমরা দেখলাম, এবং এখুনি আরো ভালো করে বুঝব, যখনই ব্যাশ কোনো একটা বিশেষ চিহ্ন পায় বা আদেশ পায়, তখন, কী ভাবে কমান্ড বা আদেশগুলোর ডিফল্ট উৎস বা মোক্ষ বদলে ফেলে। এর কিছু চিহ্ন বা কায়দা আমরা এর মধ্যেই জেনেছি। যেমন '>' বা '<' বা '|' ইত্যাদি। এই বদলে ফেলাটাই হল রিডাইরেকশন বা চালান করা। আর রিডাইরেকশনটা যখন ঘটে একটা কমান্ড থেকে আর একটা কমান্ডের কাছে, তখন আমরা বলি প্রথম কমান্ডের আউটপুট বা ফলাফলকে দ্বিতীয় কমান্ডের কাছে পাইপ করা হল বা পথ-দেখানো হল। এই 'চালান' এবং 'পথ-দেখানো' প্রতিশব্দদুটো আমরা এনেছিলাম পাঁচ নম্বর দিনের ৪ নম্বর সেকশনে। খুব একটা চলেবল তা নয়, ঠিক আছে, মানে-টা তো বোঝা যাচ্ছে। আমাদের লাগের অরিজিত অঙ্কুর বাংলার সূত্রে একটা বড় প্রতিশব্দ তালিকা বানানোর প্রস্তাব করেছে, দেখা যাক কী হয়। অঙ্কুর বাংলার প্রতিশব্দগুলো আমারও ভাল লাগেনি, কিন্তু ছেলেগুলোর কোনো দোষ নেই, যে পরিমাণ খাটছে, আর ওরা কম্পিউটারটা যেমন জানে, ভাষা বা ব্যাকরণ ততটা ভালো জানার কথাও নয় ওদের।

১২।। চিহ্নপ্রবাহের গতিপথের চালচিত্র

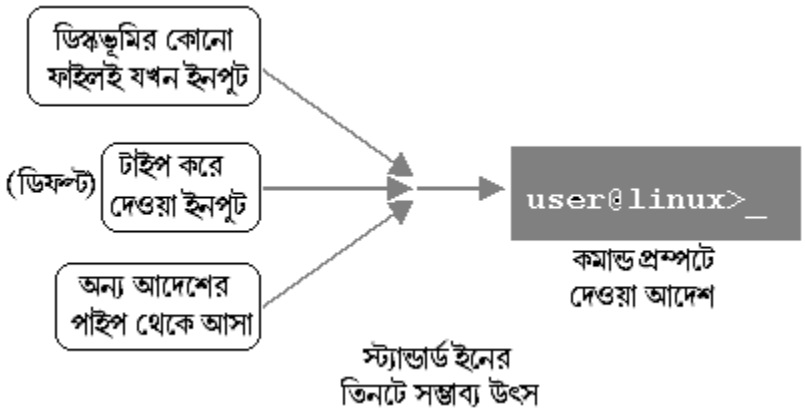
একটা আদেশ থেকে তৈরি হয় কিছু ফলাফল, কিছু চিহ্ন প্রবাহ, ক্যারেকটার স্ট্রিম, যা একটা ফাইল, তার নাম স্ট্যান্ডার্ড আউটপুট। এই স্ট্যান্ডার্ড আউট নামের চিহ্নপ্রবাহের ফাইলটার ভবিষ্যত হতে পারে তিন রকম। এক, তার ডিফল্ট গতি, অগতির গতি হল স্ক্রিন মানে কনসোল মানে মনিটর। আপনাকে চোখে গরম দেখিয়ে দিল তার কাজের ফলাফল। দুই, ফলাফল বা আউটপুটটা সরাসরি চলে গেল একটা ডিস্কফাইলে। আপনার ডিস্কভূমিতে নির্দিষ্ট কোনো একটা ফাইলে সঞ্চিত হল, সেভ করা হল। তৃতীয় গতিটা হল ত্রিশঙ্কু, মানে সরাসরি কোনো আউটপুট তৈরি হলনা, একটা কমান্ডের আউটপুট পাইপ হয়ে গেল আর একটা কমান্ডের ইনপুট হয়ে — আর একটা আদেশ যাকে আপনি দিয়েছেন ওই কমান্ড প্রম্পট থেকেই, সেই কমান্ড এবার প্রথম কমান্ডের আউটপুটটাকে নিজের মত করে চটকাবে, মানে প্রথম কমান্ডের থেকে তৈরি চিহ্নপ্রবাহের স্ট্যান্ডার্ড আউট এবার দ্বিতীয় তথা পরবর্তী কমান্ডের ইনপুট হয়ে



গেল। এবার সেই কমান্ড চটকানোর পরে আবার এই তিনটে গতি। এক স্ক্রিন, দুই ফাইল, তিন পুনঃপাইপিত হওয়া। এই গোটা প্রক্রিয়াটাকে দেখুন, এই ছবিতে দেখানোর চেষ্টা করেছি।

এখানে দেখুন, প্রথম কমান্ডের প্রক্রিয়াটাকে দেখিয়েছি স্বাভাবিক দাগে, পরের আদেশ থেকে সঞ্জাত প্রক্রিয়াটাকে দেখিয়েছি সচ্ছিদ্র দাগে, ডটেড লাইনে। আর দুটো কমান্ড প্রম্পটের মধ্যে যোগটা দেখিয়েছি ওই রকম দাগে, তার কারণ, কমান্ড প্রম্পট তো আপনি দুবার ব্যবহার করছেন না, ব্যবহার করছেন একবারই, যেমন একটু আগে আমাদের দেওয়া '`man -k CD | grep 'audio'> audio.cd.program.text`' কমান্ডটা ভাবুন। 'man' আর 'grep' আদেশদুটো তো দেওয়া হয়েছে একই সঙ্গে। এবার দেখুন 'man' আদেশটা তার ফলাফল পাঠিয়ে দিয়েছে পরবর্তী কমান্ড মানে 'grep'-এর কাছে, মানে, আমাদের ছবিতে একদম নিচের দাগটা। এটা হল পাইপ করা, বা পথ-দেখানো, যার জন্যে আমরা চিহ্ন দিয়েছি '|'। এবার 'grep' আবার তার ফলাফল থেকে তৈরি স্ট্যান্ডার্ড আউটকে

এবার পাঠিয়েছে ‘audio.cd.program.text’ নামে ফাইলের কাছে, মানে ছবিতে একদম উপরের সম্ভাবনাটায়। এটা হল রিডাইরেক্ট বা চালান করা, যার চিহ্ন দিয়েছি আমরা ‘>’। ব্যাশক্রিয়া এখানেই শেষ। ব্যাশকে আমরা অশেষ করার চেষ্টায় আরো আরো কমান্ড লাগাতে পারতাম। ধরুন ‘`man -k CD|grep 'audio'|less`’। মানে ফাইলে না-পাঠিয়ে এবার স্ট্যান্ডার্ড আউটের গতি হল তৃতীয় একটা কমান্ডের কাছে, ‘less’। তার কাছে সেটা ইনপুট, ‘less’ এবার নিজের মত করে সেই চিহ্নপ্রবাহের পিণ্ড চটকে পৌঁছে দিল স্ক্রিনে, আমরা লেস দিয়ে পাতা বাই পাতা তাকে দেখতে পেলাম। তার মানে এবার, এতক্ষণে, দুটো কমান্ডের পাইপের পথ পেরিয়ে স্ট্যান্ডার্ড আউট তার ডিফল্ট গতিপ্রাপ্তিতে পৌঁছলো।



স্ট্যান্ডার্ড আউটপুটের টার্মিনাসের সংখ্যা তিন, আর স্ট্যান্ডার্ড ইনপুটের উৎসের সংখ্যা তিন। আগের স্ট্যান্ডার্ড আউটের ছবিতে দেখুন, দিকনির্দেশটা হল কমান্ড থেকে স্ট্যান্ডার্ড আউটের দিকে চিহ্নের প্রবাহ, আর এবারের স্ট্যান্ডার্ড ইনের ছবিতে দেখুন, এখানে চিহ্নের প্রবাহটা স্ট্যান্ডার্ড ইন থেকে কমান্ডে। কমান্ড প্রম্পটে টাইপ করে দেওয়া আপনার কোনো আদেশ তার ইনপুট হিশেবে ব্যবহার করছে যে চিহ্নপ্রবাহ বা ক্যারেকটার স্ট্রিমকে, তার অন্য নাম স্ট্যান্ডার্ড ইন। আদেশের কাছে এই স্ট্যান্ডার্ড ইন আসার ডিফল্ট উপায়টা হল টাইপ করে দেওয়া ইনপুট। এছাড়া আসতে পারে আপনার ডিস্কভূমিতে ইতিমধ্যেই সেভ করে রাখা কোনো ফাইল থেকে। আসতে পারে অন্য কোনো আদেশের পাইপ মারফতও, যেখানে সেই পূর্ববর্তী সমাধিত আদেশের কাজের ফলাফল বা আউটপুট হল এই চালু বর্তমান আদেশের ইনপুট। যেমন আগের সেকশনেই ব্যবহৃত ‘`man -k CD|grep 'audio'|less`’ কমান্ডটা ভাবুন। এখানে ‘less’ আদেশের কাছে ইনপুট আসছে ‘grep’ কমান্ডের ফলাফল, আবার ‘grep’-এর কাছে এসেছিল ‘man’ কমান্ড থেকে। এই ভাবেই একটা আদেশের পাইপ বেয়ে অন্য আদেশের কাছে যাচ্ছে চিহ্নপ্রবাহ। কিন্তু যদি এই কাজটা আমরা দুটো ভাগে ভেঙে নিতাম?

ধরুন, যদি, কমান্ড দিতাম, ‘`man -k CD > cdtemp; grep 'audio' cdtemp|less`’, তাহলে কী হত? এখানে দেখুন প্রথমে আমরা ‘man’ কমান্ডের ফলাফলটাকে চালান করে দিচ্ছি একটা ডিস্কফাইলে। আচ্ছা, বলুন তো, হঠাৎ, আগের সেকশন থেকে, ফাইলকে ডিস্কফাইল বলে ডাকতে শুরু করলাম কেন? কারণটা যদি নিজেই ভেবে নিতে পারেন, বোঝা যাবে আপনার গ্নু-লিনাক্সের অভ্যেশ ভালোই চলছে। গ্নু-লিনাক্সে সবই ফাইল, তাই, ক্যারেকটার স্ট্রিম বা চিহ্নপ্রবাহ — সেটাও একটা ফাইল। কিন্তু সেই ফাইল আমাদের প্রাত্যহিক চেনা ফাইল নয়, তার আবাস সাইবার প্রক্রিয়ার ভৌতিক ভূমিতে — ভারচুয়াল করিয়া তারে করেছো এ কী সন্ন্যাসী, বাফারময় দিয়াছ তারে ছড়ায়ে? যখন আমরা ওই চিহ্নপ্রবাহকে রিডাইরেক্ট করি আমাদের ডিস্কভূমিতে ন্যস্ত ডিরেক্টরিগুলোর কোনো একটার ভিতর কোনো একটা নির্দিষ্ট নামে, নির্দিষ্ট কিছু বাইট লিখে ফেলা হয় ওই চিহ্নপ্রবাহের অন্তর্বস্তু দিয়ে, তখন সেই ফাইলটা আমাদের প্রাত্যহিক ব্যবহৃতব্য ফাইল হয়ে ওঠে। ডিস্কফাইল নাম দিয়ে এটাই বোঝাচ্ছি যে এটা ওই ধরনের কোনো সাইবারবাসী ঘনচক্কর ফাইল নয়, নিতান্তই ভাতডাল খাওয়া ডিস্কবাসী ফাইল, এমনকি পকেট হাতড়ালে এক আধটা বনগাঁ লাইনের মাস্থলি টাস্থলিও বেরোতে পারে।

হ্যাঁ, যা বলছিলাম, আগের কমান্ডটা দিয়ে ‘man’ কমান্ডের ফলাফলটাকে স্ট্যান্ডার্ড আউটের ডিফল্ট স্ক্রিনে না নিয়ে চালান করে দিচ্ছি বর্তমান কাজের চালু ডিরেক্টরিতে, মানে যে ডিরেক্টরিতে বসে কমান্ডটা দিচ্ছি সেখানে, ‘cdtemp’

নামের ডিস্কফাইলে। এর পরে দেখুন একটা ';' চিহ্ন, ঠিক যে যতিচিহ্ন হিশেবে আমরা সেমিকোলন ';' ব্যবহার করি, সেই কাজই করছে এখানে। শেষ এই কমান্ডটা আসলে পরপর দুটো কমান্ডের একটা সমাহার। পরপর কাজ করার দুটো কমান্ডকে আলাদা করে এই ';' চিহ্ন। এবার পরের কমান্ডটুকুকে আলাদা করে বোঝার চেষ্টা করুন।

```
grep 'audio' cdtemp | less
```

একটু আগে 'grep' কমান্ডটা তার ইনপুট পাচ্ছিল সরাসরি 'man' আদেশের ফলাফল থেকে। এখন আর তা নয়। 'grep' আদেশটা এখন তার ইনপুট নিচ্ছে 'cdtemp' নামের ডিস্কফাইল থেকে। সেই ফাইলের মধ্যে 'audio' শব্দটা কোন কোন লাইনে আছে সেটা খুঁজে দেখছে। তারপর সেই লাইনগুলোকে তুলে আনছে সামনে মানে আউটপুটে। সেই আউটপুটকে আমরা আবার 'less' দিয়ে পড়ছি। তার মানে, 'grep' কমান্ডকে ইনপুট হিশেবে যে উৎসই আমরা দিই না কেন — তা সে 'man' কমান্ডের ফলাফলের পাইপ হোক, বা সেই ফলাফল দিয়ে গজিয়ে তোলা ডিস্কফাইল 'cdtemp' হোক, তাতে 'grep'-এর কাজ করার কোনো পার্থক্য ঘটেনা।

স্ট্যান্ডার্ড ইনের ছবিটায় দেখুন, তিনটে উৎসের মধ্যে দুটো, উপরের আর নিচেরটা আমরা চিনলাম, মধ্যেরটা এখনো বাকি, মানে, ইনপুটটা যখন আসছে কিবোর্ড থেকে, মানে স্ট্যান্ডার্ড ইনের যেটা ডিফল্ট। কমান্ড প্রম্পটে টাইপ করে দেওয়া চিহ্নপ্রবাহ যখন কমান্ডের ইনপুটের উৎস। কিন্তু সেটাও আসলে বাকি নেই। আগেই দেখানো হয়ে গেছে। চিহ্নপ্রবাহের এই ফাইল তিনটেকে নিয়ে আলোচনা শুরু করার সময়, আজকের ১১ নম্বর সেকশনে। সেখানে আমরা কমান্ড দিয়েছিলাম, '`cat - ~/.bashrc - ~/.profile>local.conf.text`'। এই কমান্ডটা ঠিক কী কাজ করছে, কেমন ভাবে, সেই আলোচনা ওখানেই আছে। 'cat' কমান্ডটা একবার তার ইনপুট একবার পাচ্ছে '-' চিহ্ন দিয়ে দেখানো কিবোর্ড ইনপুট থেকে, একবার পাচ্ছে '~/.bashrc' ইত্যাদি ডিস্কফাইল থেকে। কিবোর্ড বা ডিস্কফাইল যে উৎস থেকেই আসুক, তাতে ব্যাশের কিছু এসে যাচ্ছেনা।

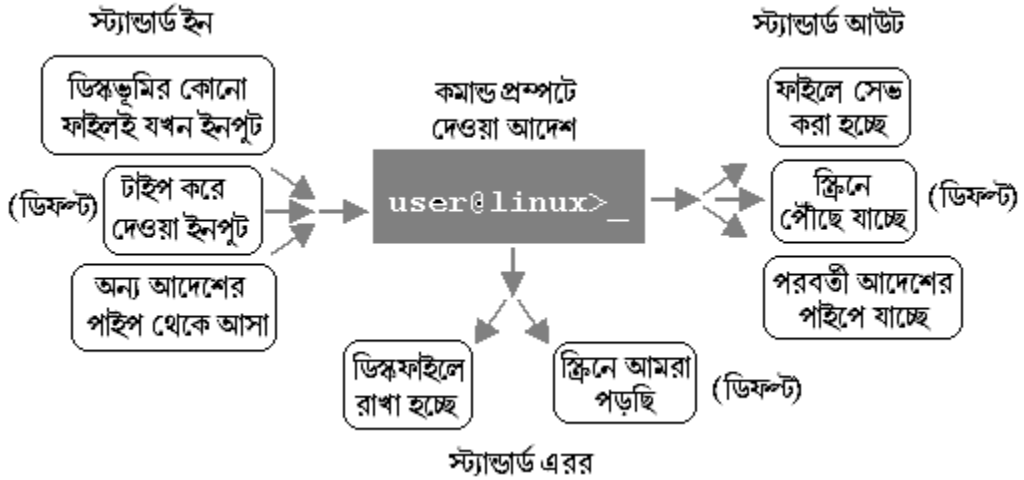
এই স্ট্যান্ডার্ড ইন আর আউটেই কিন্তু ব্যাশচরিত শেষ হয়না, আদেশ পালন করতে গিয়ে বহু রকম প্রমাদ ঘটে, মানে, আদেশ পালনে সিস্টেমের তথা ব্যাশের অপারগতা। খুব সহজটাই ভাবুন, এমন একটা ফাইল আপনি কনক্যাটেনেট করতে, 'cat' করতে, বললেন, যে নামের কোনো ফাইলই ওই ডিরেক্টরিতে নেই, যে ডিরেক্টরিতে দাঁড়িয়ে আপনি কমান্ডটা দিয়েছেন। ধরুন, এই আগের সেকশনেই আমরা যে 'cdtemp' ফাইলটা বানিয়েছিলাম, সেটা স্বাভাবিকভাবেই, নামে যেমন বোঝা যাচ্ছে, একটা অস্থায়ী বা টেম্পোরারি ফাইল, তাই কাজ মিটে যাওয়ার পরেই আমরা উড়িয়ে দিয়েছি। এবার, এখন যদি হঠাৎ করে কমান্ড দিই, '`cat cdtemp`', স্বাভাবিকভাবেই, ব্যাশ সেই নামের কোনো ফাইল আপনার হোম ডিরেক্টরিতে খুঁজে পাবেনা, ফাইলটাই তো আর নেই। তাহলে ব্যাশ এখন কী করবে? সে আপনাকে প্রমাদ বার্তা বা এরর মেসেজ পাঠাবে। ব্যাশ আপনার কনসোলে ফুটিয়ে তুলবে,

```
cat: cdtemp: No such file or directory
```

এক একটা আদেশ পালনে সিস্টেমের এক এক ধরনের অপারগতার জন্যে এক এক রকম প্রমাদ বার্তা। এই প্রমাদ বার্তাটা আপনি একটা ফাইলেও পাঠাতে পারেন, তারও উপায় আছে, সে কথায় আসছি আমরা। আপাতত এই কমান্ডটা দেখুন, '`cat cdtemp 2>errorfile`'। এর আগে আমরা যখন শুধু ক্যাট করার কমান্ডটা দিচ্ছিলাম, আমাদের যে প্রমাদবার্তাটা ব্যাশ স্ক্রিনে দেখাচ্ছিল সেই বার্তাটাকেই এখন চালান করে দিচ্ছে 'errorfile' নামের ডিস্কফাইলে। চালান করার ওই চিহ্নটা, '>', আমরা চিনি, শুধু এখানে ওই '2' সংখ্যাটা আমরা চিনিনা, দুনম্বরিটা চিনে যাব, ক্রমে আমরা ব্যাশিক্ষিত হচ্ছিনা? এবার যদি আমরা ওই ডিস্কফাইলটাকে স্ক্রিনে পড়ি, '`cat errorfile`' কমান্ড দিয়ে, ঠিক সেই এরর মেসেজটাই পাব, যেটা এর আগে ব্যাশ আমাদের সরাসরি স্ক্রিনে দেখাচ্ছিল।

ব্যাশের এই প্রমাদ বার্তাই হল চিহ্নপ্রবাহের তিন নম্বর ফাইল, স্ট্যান্ডার্ড এরর। ছবিতে দেখুন, উপরের বাঁদিকের স্ট্যান্ডার্ড ইন, বা ডানদিকের স্ট্যান্ডার্ড আউট আমাদের চেনা, এর সঙ্গে এসেছে স্ট্যান্ডার্ড এরর, যার আবার দুটো গতি, হয় স্ক্রিনের ডিফল্ট, নয়ত রিডাইরেক্ট করা ডিস্কফাইল। এইমাত্র যেটা আমরা করে দেখলাম। ব্যাশ আমাদের প্রমাদবার্তা পাঠিয়েছিল, কারণ আমরা '`cat cdtemp`' কমান্ড দিয়ে 'cdtemp' নামের এমন একটা ফাইল 'cat' করতে বলেছিলাম, যে ফাইলটাই নেই। তার থেকে তৈরি হয়েছিল প্রমাদ বা এরর। যে বার্তা বা মেসেজটা আমরা

পরে ‘errorfile’ নামের ফাইলে পাঠিয়েছিলাম, পুরোনো ‘cat cdtemp’ কমান্ডটার সঙ্গে ‘2>errorfile’ বলে একটা বাড়তি অংশ যোগ করে।



এই ‘2’ সংখ্যাটাকে বলে ফাইল ডেসক্রিপ্টর। ফাইল ডেসক্রিপ্টর দিয়ে আমরা ইন, আউট আর এরর — এই তিন স্ট্যান্ডার্ড ক্যারেকটার স্ট্রিম ফাইলকে চিনে থাকি। তিনটের জন্যে তিনটে আলাদা আলাদা সংখ্যা। ‘0’ মানে স্ট্যান্ডার্ড ইন, ‘1’ মানে স্ট্যান্ডার্ড আউট আর ‘2’ মানে স্ট্যান্ডার্ড এরর। চালান বা রিডাইরেক্ট করার চিহ্ন ‘>’-এর সঙ্গে এই তিনটে সংখ্যার যে কোনোটাকে লাগানো মানে সেই চিহ্নপ্রবাহটাকে রিডাইরেক্ট করা। যেখানে তার ডিফল্টে যাওয়ার কথা, সেখানে যেতে না-দিয়ে আমরা অন্য কোথাও নিয়ে যাচ্ছি তাকে, চালান করছি, রিডাইরেক্ট করছি। এইমাত্র আমরা যে ‘2>errorfile’ অংশটা যোগ করলাম, তার মানে, এরর বার্তাটাকে তার ডিফল্ট গতি মানে স্ক্রিনে যেতে না-দিয়ে আমরা ঠেলে দিচ্ছি একটা ফাইলে, যার নাম ‘errorfile’। তার আগে অব্দি প্রমাদবার্তাটা আমরা স্ক্রিনে পাচ্ছিলাম। এখানে স্ট্যান্ডার্ড এরর বোঝাতে ‘2’ ফাইল ডেসক্রিপ্টরটা ব্যবহার করতে হল, কিন্তু, ‘0’ বা ‘1’-এর বেলায় সচরাচর করতে হয়না, কারণ সেটাই ডিফল্ট। ধরুন আমরা চালু ডিরেক্টরির সমস্ত ফাইলের তালিকা বানাচ্ছি ‘ls’ কমান্ডের স্ট্যান্ডার্ড আউটকে রিডাইরেক্ট করে, ‘ls > filelist’ কমান্ড দিয়ে। এখানে আমরা যা আদতে করছি, তা হল, ‘ls 1> filelist’। কিন্তু ‘1’-টা আলাদা করে উল্লেখ করতে হয়নি, কারণ ওটাই ডিফল্ট। কখনো কখনো সেটা করতে হয় বৈকি, একাধিক প্রবাহকে একই সঙ্গে একটা ফাইলে রিডাইরেক্ট করতে হলে। কিন্তু সে ব্যাশের অনেকটা অগ্রবর্তী কমান্ডের ব্যবহারে। খুব যদি কৌতূহল হয়, তো, খেজুরগাছে হাঁড়ি বাঁধো ম্যান, ম্যান পড়ার নেশা করো ম্যান।

১৩।। কমান্ডের মধ্যে কমান্ড

একটা কমান্ডের তৈরি আউটপুট যখন সরাসরি আর একটা কমান্ডের ইনপুট হয়ে আসে, সেটা আমাদের চেনা। আমরা সেই পাঁচ নম্বর দিন থেকে আজ অব্দি অজস্রবার ব্যবহার করে আসছি। একে আমরা ডাকি পাইপ করা বা পথ-দেখানো। যার চিহ্ন ‘|’। বারবার করেছি ব্যবহার। এর আলোচনায় আর যাচ্ছিনা। এই পাইপ ছাড়া আরো কয়েকভাবে একটা কমান্ডের মধ্যে আর একটা কমান্ড ব্যবহার করা যায়। আজকের আলোচনার ৯ নম্বর সেকশনের একদম শেষে এসে আমরা এর একটার কথা উল্লেখও করেছিলাম, ব্যাককোট বা ‘`’। এই চিহ্নটা পাবেন কিবোর্ডের একদম বাঁয়ে উপরে, ঠিক ‘<Esc>’ বা ‘এসকেপ’ চাবিটার নিচেই। আমরা এই চিহ্নটার আলোচনা এনেছিলাম ‘writefiles’ নামে ব্যাশস্ক্রিপ্টের সূত্রে। লাইনটা ছিল ‘c=`expr $c + 1`’। এই স্ক্রিপ্টে আমরা একটা কাউন্টার ভ্যারিয়েবল ব্যবহার করেছিলাম, যার নাম ‘c’, যার আলোচনা আমরা নয় নম্বর দিনেই করেছি, ২.২ নম্বর সেকশনে, স্ক্রিপ্টের লাইন ধরে ধরে আলোচনায়, একবার দেখে নিন। এই লাইনটার মোদ্দা কাজ হল ‘c’ নামের ভ্যারিয়েবলটার মান প্রতিবার ঘোরার সঙ্গে সঙ্গে এক করে বাড়ানো। ঘোরা বলতে ‘writefiles’ স্ক্রিপ্টে দেখুন, একটা একটা করে ফাইল ক্যাট করে তুলে আসা। এর আগের লাইনটাই দেখুন, ‘cat $i>> $f’, যেখানে ও একটা ফাইলকে তুলে দিচ্ছে। আর তার পরেই ‘c’-এর মান এক বাড়িয়ে দিচ্ছে। কেন বাড়াচ্ছে? কারণ, তার আগের লাইনে যেখানে প্রতিটি ফাইল

ক্যাট করার আগে একটা করে হেডিং বানাচ্ছে, সেখানে ফাইলের নম্বরটা লিখে দিচ্ছে ব্যাশ। স্ক্রিপ্টটার গোড়াতেই দেখুন অ্যাসাইন করা আছে, 'c=1'। তার মানে প্রথম ফাইলের হেডিং-এ এটাই আসবে, ফাইলটা ক্যাট করার পর 'c'-এর মান এক বাড়িয়ে দেবে, তার মানে পরের ফাইলে মানটা আসবে দুই, এই ভাবে বাড়তে থাকবে। এই কাউন্টার ব্যাপারটা প্রোগ্রাম লেখার একদম গোড়ার একটা খুব প্রাথমিক তরকিব।

এবার লাইনটাকে বোঝার চেষ্টা করা যাক — কী ঘটছে লাইনটায়। 'expr' হল একটা অঙ্ক কষার কমান্ড। সিস্টেমেই থাকে। আপনার প্রিয় ম্যানুয়ালদের এক নম্বর সেকশনেই দেখুন দেওয়া আছে এর ইতিবৃত্ত। প্রথমে দুই ব্যাককোট চিহ্নের মধ্যের অংশটাকে ভাবুন। 'expr $c + 1'। এটা নিজেই আলাদা একটা কমান্ড। অ্যাসাইন করার কমান্ড। আমরা আগেই বলেছি '=' চিহ্নটা অ্যাসাইন বা ধার্য করে। এখানে গোটা প্রক্রিয়াটা ভাবুন। প্রথমে, স্ক্রিপ্টের একদম গোড়াতেই আপনি ব্যাশকে জানিয়েছেন, 'c=1', অর্থাৎ 'c'-এর মান ধার্য করো '1'। অর্থাৎ এইমুহূর্তে ব্যাশ জানে 'c' হল '1', মেশিনের স্মৃতিপটে 'c' নামক ভ্যারিয়েবলের জন্যে বরাদ্দ করা জমিতে সে লিখে রেখেছে '1' সংখ্যাটা। ভ্যারিয়েবলের এই অ্যাসাইন করার ব্যাপারটা তো আমাদের পরিচিত। এবার সে প্রথম ফাইল ক্যাট করে দিল। করার পরে এল আমাদের উদ্দীষ্ট লাইনে। এখানে সে পাচ্ছে 'expr $c + 1'। 'expr' কমান্ডটা নিছক অঙ্ক কষে। তাকে আমরা বললাম, তুমি '$c' মানে 'c' ভ্যারিয়েবলের মানের সঙ্গে '1' যোগ করো। যোগ করে সে কী পাবে? তার কাছে আছে, 'c'-এর মান হল '1', এর সঙ্গে সে আরো '1' যোগ করবে, তার মানে, যোগফল হবে '2'।

ধরুন ওই স্ক্রিপ্টের ভিতরে নয়, বাইরে, একদম ব্যাশ কমান্ড প্রম্পটে এই গোটা কাজটা আমরা করছি। যদি পরপর এই দুটো কমান্ড দিই আমরা, প্রথম লাইনে একটা ভ্যারিয়েবল অ্যাসাইন করি, 'c=1', আর তার পরের লাইনে দিই, 'expr $c + 1', তাহলে ব্যাশ স্ক্রিনে ফুটিয়ে তুলবে, '2'। মানে অঙ্কটা সে কষে ফেলল, 'expr' কমান্ডটা দিয়ে যে অঙ্কটা তাকে কষতে বললাম আমরা। কিন্তু আমাদের কাজ এখনো শেষ হয়নি। এই মুহূর্তে আমরা ব্যাশ প্রম্পটে যদি দিই, 'echo $c', ব্যাশ আমাদের কী দেখাবে? ব্যাশ দেখাবে '1', কারণ 'c' ভ্যারিয়েবলের মান ওটাই, ব্যাশের কাছে তাই দেওয়া আছে। এবার, আমাদের উদ্দীষ্ট ওই গোটা লাইনটা ব্যাশের কমান্ড প্রম্পটে আদেশ আকারে দেওয়া যাক, 'c=`expr $c + 1`'। এবার আর একবার 'echo $c' দিয়ে দেখুন। এবার দেখবেন, ব্যাশ ফুটিয়ে তুলল, '2'। অর্থাৎ, ব্যাককোট বা '`' চিহ্নের মধ্যের যে অংশটা, যার মান '2' আমরা এইমাত্র দেখেছি সেটাকে অ্যাসাইন করে দেওয়া হয়েছে 'c' ভ্যারিয়েবলের জন্যে বরাদ্দ করা জমিতে। তাহলে, '=' চিহ্ন দিয়ে ভ্যারিয়েবল অ্যাসাইনমেন্ট কমান্ডের মধ্যে আমরা আর একটা কমান্ড 'expr'-কে ব্যবহার করে নিতে পারলাম এই ব্যাককোট চিহ্নের দৌলতে।

আমাদের স্ক্রিপ্টটাকে এবার ভাবুন, প্রতিবার, ক্যাট করার পরের লাইনেই, ব্যাশ কী করছে? 'c' ভ্যারিয়েবলের সেই মুহূর্তের মান পড়ে নিচ্ছে মেমরি থেকে, সেই মানটার সঙ্গে 'expr' কমান্ড ব্যবহার করে '1' যোগ করে নিচ্ছে, তারপরে এই যোগফলটাকে অ্যাসাইন করে দিচ্ছে 'c' ভ্যারিয়েবলের জন্যে বরাদ্দ করা জমিতে। মানে প্রতিবার 'c' ভ্যারিয়েবলটা এক করে বেড়ে যাচ্ছে। তাই প্রত্যেকটা ফাইলের হেডিং-এ আসা সংখ্যাটা আগের ফাইলের সংখ্যাটার থেকে এক বেশি হচ্ছে। আমরা ঠিক যেভাবে গুনি, সেই গোনার প্রক্রিয়াটা আমাদের ব্যাশ স্ক্রিপ্টিং ভাষা দিয়ে আমরা ব্যাশের হাতে দিয়ে দিচ্ছি। নয়মান সাহেবের 'CC' বা কেন্দ্রীয় নিয়ন্ত্রণ এখন মেশিনের ভিতরে চলে গেছে, ক্যালকুলেটরে যা ছিলনা।

আদেশ বদল বা কমান্ড সাবস্টিটিউশনের এই কাজটা আমরা আর এক ভাবেও করতে পারতাম। আমাদের ইতিমধ্যেই চেনা '$' চিহ্ন দিয়ে। আমাদের 'writefiles' স্ক্রিপ্টে যদি 'c=`expr $c + 1`' লাইনটা ইম্যাক্স দিয়ে মুছে, তার জায়গায় এই লাইনটা যোগ করে দিই, 'c=$(expr $c + 1)', স্ক্রিপ্টটা একই ভাবে কাজ করবে। নিজের মুন্ডুর ভিতরকার প্রক্রিয়াগুলোর 'CC' যদি আপনি অন্য কারো হাতে বকলমা না-দিয়ে থাকেন, তাহলে, এখানে কী ঘটছে সেটা আপনি নিজেই বুঝে নিতে পারবেন। শুধু মাথায় রাখবেন, '$' মানে মান। প্রথমে করছি '$c' অর্থাৎ 'c'-এর মান, তার সঙ্গে 'expr' দিয়ে '1' যোগ করছি। এবার এই গোটাটাকে দেখছি একটা ভ্যারিয়েবল হিশেবে, যার মান বার করছি এই গোটাটা দিয়ে — '$(expr $c + 1)'। এবং সেই মানটাকে অ্যাসাইন করে দিচ্ছি 'c' ভ্যারিয়েবলের জন্যে বরাদ্দ করা জমিতে তার পুরোনো মান মুছে দিয়ে।

কমান্ডের মধ্যে কমান্ড টেনে আনার শেষ প্রক্রিয়াটা বেড়ে মজার। কমান্ডটার নাম 'tee', এতে গাছেরটা খাওয়া যায়, এবং একই সঙ্গে, একটুও না-থমকিয়ে বা না-চমকিয়ে তলারটাও কুড়োনো যায়। আমাদের আজকের আলোচনার ২

নম্বর সেকশনে '.bashrc' ফাইল থেকে কয়েকটা লাইন তুলে দিয়েছিলাম আমরা, সেখানে বলেছিলাম শেষ চারটে লাইন এই মুহূর্তে খ্যামা দিন। সেই লাইন চারটের গোটাটা বোঝার ক্যালি আমাদের এখনো হয়নি। কিন্তু মধ্যের একটা লাইন এবার পেড়ে ফেলা যাক।

```
echo -e '\n***\n'>>F;/usr/bin/fortune -s | tee -a F | cat
```

এই লাইনটায় ব্যবহার হয়েছে মোট চারটে কমান্ড, 'echo', 'fortune', 'tee', এবং 'cat'। 'echo' কমান্ডের পরে '-e' অপশনটার মানে আমরা জানি, যাতে '\n' দেখলে একো সত্যিই একটা করে লাইন বাদ দেয়। একোকে আদেশ দেওয়া হয়েছে, তুমি 'F' নামের ফাইলে প্রথমে যোগ করো একটা ফাঁকা লাইন, তার পরের লাইনে তিনটে তারকাচিহ্ন বা অ্যাসটেরিক্স '***' যোগ করো, তার পরে ফের একটা ফাঁকা লাইন যোগ করো। খেয়াল করুন, এখানে আমরা রিডাইরেকশনের '>' ব্যবহার না-করে, অ্যাপেন্ড বা যোগ করার '>>' চিহ্ন দিয়েছি। এর মানে আমরা জানি, শুধু '>' চিহ্নে ডিরেক্টরিতে থাকা পুরোনো 'F' ফাইল মুছে নতুন একটা 'F' নামের ফাইল খুলে নেবে ব্যাশ, কিন্তু '>>' থাকায় সেটা না-করে, পুরোনো ফাইল থাকলে তাতে যোগ করে দেবে, আর না-থাকলে ওই নামের একটা ফাইল বানিয়ে নেবে। এর পরে আছে একটা ';' চিহ্ন। তার পরে পরের কমান্ড '/usr/bin/fortune -s'। কমান্ড প্রম্পটে দিলে একটা করে বাণী শোনায় ফরচুন, এর কথা তো আগে বারংবার বলেছি। শুধু '-s' অপশনটা ফরচুনকে জানায়, একটু ছোট করে কাকা, শর্টে মেরো। আর '/usr/bin' অংশটা হল 'fortune' নামের বাইনারির ঠিকানা। তারপরে একটা পাইপ করার চিহ্ন '|'। তার মানে ফরচুন বাণীটাকে আমি সরাসরি স্ক্রিনে না-দেখে পাইপ করে দিচ্ছি। এটা খুব চেনা প্রক্রিয়া। ধরুন এরকম যদি কমান্ড দিই আমরা, 'fortune -s >> F' — এতে ফরচুন বাণীটা স্ক্রিনে না-দেখে আমরা 'F' ফাইলে চালান করে দিই। হয় স্ক্রিন নয় ফাইল। দুটো এক সঙ্গে হয় না। কিন্তু এই পাইপ চিহ্নের পরে 'tee' কমান্ডটা এবার দিচ্ছে সেই মজা, কেকও খাবো, আবার তার ডোটাও রয়ে যাবে। 'tee -a F' কমান্ডটা ব্যাশকে বলে দিচ্ছে, প্রথমে তুমি 'fortune' কমান্ডের কাছ থেকে পাইপ করে যেটাকে পেয়েছ সেটাকে 'F' ফাইলে যোগ করো, ওই '-a' অপশনটার মানে হল অ্যাপেন্ড বা যোগ করা। কিন্তু মজাটা দেখুন, এটা 'tee' না-হয়ে 'cat' হলে কমান্ডটা এখানেই শেষ হয়ে যেত, নতুন করে পাইপ করার আর কিছু থাকত না, স্ট্যান্ডার্ড আউটটা পৌঁছে যেত 'F' ফাইলে। কিন্তু এখানে তা হলনা, 'tee' যে চিহ্নপ্রবাহটুকু যোগ করল 'F' ফাইলে, সেটাকে আবার রেখেও দিল। পাইপ করে দিল কমান্ডের একদম শেষ অংশে, 'cat'। তার মানে, এবার আমরা তাকে স্ক্রিনে পড়তে পেলাম। একই সঙ্গে 'F' নামের ফাইলে ফরচুন বাণীগুলো যোগ হতে থাকল, এবং সেগুলো লগ-ইন করা মাত্র আমি স্ক্রিনে পড়তেও পেলাম। আপনাদের কারোর যদি দরকার পড়ে, এভাবে জমানো অফেনসিভ ফরচুন বাণী আমার ফাইল করে রাখা আছে, সেটা আপনাদের দিতে পারি, মেল করবেন।

এইবার আমরা ঢুকব ব্যাশ নিয়ে আমাদের আলোচনার একদম শেষ অংশে। ব্যাশের কয়েকটা অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটার নিয়ে কথা বলব, কয়েকটা অতিচিহ্ন বা মেটাক্যারেকটার, একটু লুপ ছোঁব, তার পরেই আমাদের খেল খতম, জিএলটি ইশকুল পাঠমালা হজম। আমি ছুটি পাব নভেম্বর এক থেকে আজ আঠেরোই মার্চ অব্দি আমার এই ছোটার হাত থেকে। হার্ডডিসকের কারণে কয়েকটা দিন ছেড়ে দিলে আর কলেজের কাজগুলো ছেড়ে দিলে আমি উনিশ ঘন্টার বেশি নষ্ট করিনি এই কটা মাসে।

১৪।। অবস্থানগত নিয়ন্তা — এবং তাদের বদলানো

এর আগে ছয় নম্বর দিনে এবং আজকেরও গোড়ার দিকে আমরা ব্যাশের আভ্যন্তরীন ডিফল্ট রকমে সিস্টেম ভ্যারিয়েবলগুলোকে চেনার কাজে 'set' বলে একটা কমান্ড ব্যবহার করেছি। এটা 'set' কমান্ডের দিক থেকে দেখলে, অত্যন্ত অপমানজনক। ধরুন একটা তিমির একটা ডানাই যদি তিমিটার পরিচিতি হয়ে দাঁড়ায়। কিম্বা, আপনি বললেন, মাছি নিয়ন্ত্রণে লিটলবয় এবং ফ্যাটম্যানের কোনো তুলনা হয়না, হিরোশিমা নাগাসাকিতে অন্তত বছরখানেক কোনো মাছি ছিলনা। এর আগেও আমরা ব্যাশের আভ্যন্তরীন বিল্ট-ইন কাজগুলোর কথা এনেছি। ওরফ বা অ্যালিয়াস, কিম্বা, রপ্তানি বা এক্সপোর্ট ইত্যাদির আলোচনার সূত্রে। ব্যাশের ম্যানুয়ালেই পেয়ে যাবেন 'set' নামের বিল্ট-ইন কমান্ডের এবং অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটারগুলোর কথা। সেট আদেশটাকে আপনি একা একলা কমান্ড প্রম্পটে দিয়ে এন্টার মারলে কী হয়, সেটা দেখেছি আমরা। সমস্ত চালু ব্যাশ ভ্যারিয়েবলের, ফাংশনের একটা তালিকা

তৈরি করে দেয় সামনে। কিন্তু এই সেট দিয়ে একটা বিপুল কাজ করা যায়, নিজের প্রয়োজনমাফিক কিছু নিয়ন্তা বা প্যারামিটারকে ব্যাশের কাছে পৌঁছে দেওয়া, এর পরে আপনার কাজে বারবার যে নিয়ন্তাগুলোকে নিজের খুশি মত ব্যবহার করতে পারবেন।

প্রথমে একটা অর্থহীন উদাহরণ দিয়ে ব্যাপারটা বুঝে নেওয়া যাক। একটা কমান্ড দিন

```
set Arab kafela is a procession of man camel struggle and broken dreams
```

এন্টার মারুন। দেখুন, কমান্ড প্রম্পট ফেরত এসে গেছে, তার মানে যা কিছু ঘটার ঘটে গেছে। এটা ব্যাশেষত্ব। মানে ব্যাশের বিশেষত্ব। ও একজন আদ্যোপান্ত নীরব কর্মী। যে কাজটা আপনি করতে দিয়েছেন, সেটা নীরবে করে দেবে। রব তৈরি করবে একমাত্র তখনই যখন কোথাও কোনো একটা প্রমাদ ঘটবে। স্ট্যান্ডার্ড এররকে তার ডিফল্টে মানে স্ক্রিনে শো করে দেবে তখন। এখানে ব্যাশ কোনো উচ্চবাচ্য করল না, মানে, যা করার করে দিয়েছে। কিছু একটা ঘটে গেছে মেশিনের ইলেকট্রনিক পেটে। এখুনি যা মালুম পাওয়া যাবে। এবার কমান্ড দিন

```
echo `whoami` $3 $4 $8
```

হুবহু যা লেখা তা যদি টাইপ করে দিতে পারেন, আপনার আইডেন্টিটি ক্রাইসিসের একদম হাতে হাতে একটা সমাধান পেয়ে যাবেন। পেলেন? খুব বেশি খচে যাওয়ার কোনো কারণ নেই, এমন নয় যে ফাঁকতালে আমি আপনাকে গাল দিয়ে নিলাম। ভাবুন, এটা এখানে লেখার আগে করে দেখে নিতে গিয়ে হুবহু ওই আত্মজ্ঞান আমারও পেতে হয়েছিল। আর এত চটার কী আছে? 'রক্তবরণ-মুগ্ধকরণ-নদীপাশে-যাহা-বিঁধিলে-মরণ' — তিনিই তো বলেছিলেন, 'উট উটঅ উঠঅ: সে তো মিস্টিরিয়াস অ্যানিমাল মশাই' (কার্টসি জটায়ু)।

এবার দেখা যাক, এখানে কী ঘটছে। আচ্ছা, 'whoami' কমান্ডটার কথা আগে কোথাও বলেছি? মনে পড়ছে না, আর খোঁজা তো অসম্ভব। এই ইতিমধ্যেই এক লাখ চৌষট্টির জগদ্দলে। না, এই 'whoami' কোনো গভীর দার্শনিক আত্মজিজ্ঞাষা নয়, নিতান্তই একটা চালু ব্যাশের চালু ব্যবহারকারীর ইউজার নেম ফুটিয়ে তোলার কমান্ড। সেটাকে আমরা দিয়েছি দুটো ব্যাককোটের মধ্যে। মানে, কমান্ড সাবস্টিটিউশন, আদেশ-বদল। কমান্ডের মধ্যে কমান্ডের সেই চক্কর, আমরা 'whoami' কমান্ডের ফলাফলটাকে এখানে 'echo' কমান্ডের মধ্যে টেনে আনছি। এবার তার পরের অংশটা, মানে, '$3 $4 $8' — এটা আমাদের সেট কমান্ডের অবদান। তার আগেই যে লাইনটা আমরা সেট কমান্ডে দিয়েছিলাম, সেই লাইনটায়, 'set' শব্দটার পর থেকে গুনুন, এক নম্বর শব্দ হল 'Arab', দু নম্বর 'kafela'। এরকম করে তিন চার আর আট নম্বর শব্দ তিনটে কী কী সেটা দেখে নিন। এবার আপনার আত্মজ্ঞানের রহস্যভেদ হল? আসলে 'set' ঠিক এটাই করে। তার পরে যে শব্দগুলো আসছে সেগুলোকে স্পেসে স্পেসে আলাদা করে এক দুই তিন চার — এই ভাবে পরপর সংখ্যায় অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটার হিশেবে তুলে রাখে। এবার যখনই দরকার পড়ে, আমরা তাদের '$1', '$2', '$3' ইত্যাদি নামে ডাকি, এবং তাদের বৈদ্যুতিন অন্ধকার থেকে ডাকিনীমন্ত্রে জাগিয়ে তোলে ব্যাশ।

এবার এর থেকে একটু কম অর্থহীন একটা কাজ করানো যাক ব্যাশকে দিয়ে। আর তাতে, আমাদের '~/.bashrc' ফাইল বদলানোটাও একটু অভ্যেশ হবে। এই তিনটে লাইন আপনার '~/.bashrc' ফাইলের শেষে যোগ করে দিন

```
set $(date)
echo Hello Dear $(whoami)
echo Today is $1-day, $2, $6, and the time is: $4
```

এবার একবার লগ-আউট লগ-ইন করুন, বা, জ্যান্ত রকমে মানে ইন্টারাক্টিভলি একটা ব্যাশ শেল চালু করুন, ব্যাশের ভাষায় ইনভোক। মানে জাস্ট কমান্ড প্রম্পটে 'bash' লিখে এন্টার মারুন। এটা আমি করায় ব্যাশ আমায় দেখাল

```
Hello Dear dd
Today is a Fri-day, Mar, 2004, and the time is: 10:58:09
```

এর মধ্যে একটা জিনিষ তো গত সেকশনেই শিখেছি, ব্যাককোট বা '`' চিহ্নের জায়গায় '$()' দিয়ে আর এক রকমের আদেশ-বদল বা কমান্ড সাবস্টিটিউশন। আমরা এই একই ফল পেতাম যদি আগের তিনটে লাইন এরকম লিখতাম

```
set `date`
```

```
echo Hello Dear `whoami`
echo Today is $1-day, $2, $6, and the time is: $4
```

এর তিন নম্বর লাইনটাকে বোঝার চেষ্টা করার আগে একবার 'date' কমান্ডটা ব্যবহার করে দেখে নিন, কী ফলাফল তৈরি হয়। আমি কমান্ড লাইনে 'date' লিখে এন্টার মারায় ব্যাশ স্ক্রিনে দেখাল

```
Fri Mar 19 10:59:11 IST 2004
```

'date' কমান্ডটা ঠিক এটাই করে, দিন তারিখ সময় তুলে দেয়। এবং লগ-ইন করার মুহূর্তে ব্যাশ যা পড়েছিল তার থেকে আমি 'date' কমান্ডটা দেওয়ার মধ্যে দেখুন এক মিনিট দু সেকেন্ড সময় চলে গেছে। প্রত্যেকবারই লগ-ইন করার সময় ব্যাশ একবার করে '~/.bashrc' ফাইল থেকে এই তিন লাইন কমান্ড পড়বে, এবং সেই অনুযায়ী ফুটিয়ে তুলবে। যেমন আমরা আগেই বলেছি, করার কথা ব্যাশের। এবং ব্যাশ কী রকমে অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটারগুলো পড়ল, সেটা খেয়াল করুন। ব্যাশ নিজে 'date' কমান্ডটা দিয়ে যা পেয়েছিল, সেটা হল 'Fri Mar 19 10:58:09 IST 2004'। এবার স্পেস বা ফাঁকা জায়গার যতিটা মাথায় রাখুন। তাহলে ব্যাশের কাছে এখন '$1' মানে 'Fri', '$2' মানে 'Mar', '$3' মানে '19', '$4' মানে '10:58:09', '$5' মানে 'IST', এবং '$6' মানে '2004'। এবার জাস্ট আমাদের দেওয়া আদেশের তৃতীয় লাইনের দিকে তাকান, এবং দেখুন, ব্যাশ আর কিছুই করেনি, শুধু, '$1', '$2', '$6' এবং '$4' নামের অবস্থানগত নিয়ন্তাদের জায়গায় তাদের নিজের নিজের মান ভরে দিয়েছে। এবং ভাবুন, এই ভাবে পাওয়া অবস্থানগত নিয়ন্তাদের কত রকম ভাবে ব্যবহার করা যায়। যান করতে শুরু করুন, কোনো বাধা নেই, শুধু আপনার কল্পনাশক্তি এবং ব্যাশ ম্যানুয়ালই এর লিমিট। এই 'set' কমান্ড দিয়ে পাওয়া অবস্থানগত নিয়ন্তাদের আয়ু কত? পরের বার 'set' মেরে নতুন নিয়ন্তা বানানোর আগে অব্দি। অথবা, আপনি লগ-আউট করা অব্দি। প্রায় এই একই রকম ভাবে এই ধরনের কাজে ব্যবহারের আর একটা বেড়ে জিনিষ আছে ব্যাশে। তার নাম বিশেষ ব্যাশ চল বা স্পেশাল ব্যাশ ভ্যারিয়েবল।

১৫।। বিশেষ ব্যাশ ভ্যারিয়েবল

এই বিশেষ ব্যাশ ভ্যারিয়েবল বলার আগে একটা মাপ চেয়ে নেওয়ার আছে। এই বইয়ে বারবার বলেছি আমার কম্পিউটার করার স্বশিক্ষিত মানে অশিক্ষিত রকমের কথা, কোনো ভালো জায়গায় ভালো করে কিছু শেখা হয়নি, সেটা লাগের ছেলেমেয়েদের দেখলে বুঝতে পারি। এই দৌড়ে তো হলনা, পরের চান্সে ট্রাই করা যাবে, তবে ততদিনে আর কোনো হিজল-বট-জারুল এই বাংলায় থাকবে কিনা কে জানে। ব্যাশ বা শেল আমার নিজের কাছে বেশ উত্তেজক লাগে। সেই উত্তেজনাটা আমার মধ্যে চারিয়ে গেছিল স্টিফেন প্রাটার 'অ্যাডভান্সড ইউনিক্স এ প্রোগ্রামারস গাইড' (Stephen Prata: 'Advanced Unix — A Programmer's Guide') বলে একটা বই থেকে। সত্যি বলছি, একটুও বাড়িয়ে বলছি না, কলেজ থেকে ফেরার সময় আমি নিজে নটা-পাঁচের বনগা লোকালের অন্যের-কান-চুলকে-দেওয়া ভিড়ে দাঁড়িয়েও পড়ে দেখেছি, বেশ পড়া যায়, এত ভালো বই। ট্যানেনবম-এর মডার্ন অপারেটিং সিস্টেমের মত ক্রাইম থ্রিলার না হলেও মোটামুটি সাইফি সিনেমা বলে চালানো যায়। আমার কম্পিউটার ভাবনার যেটুকু যতটুকু — তা প্রায় আগাগোড়াই এই দুটো বইয়ের কল্যাণে তৈরি। কিন্তু এখানেই একটা ব্যথা আছে। এ দুটো বইয়ের একটাও গ্নু-লিনাক্সের জন্যে কাস্টম-বিল্ট নয়। ট্যানেনবমে তবু গ্নু-লিনাক্সের কোনো কোনো আলোচনা আছে। আর প্রাটার বইটার ভারতীয় সংস্করণই ছিয়াশির, মানে, লিনাস তখনো তার কারনেলই নামায়নি নেটে, পাঁচ নম্বর দিনের গোড়ার ক্রোনলজিটা দেখুন। আর প্রাটার এই বইটা বর্ন শেল নিয়ে লেখা। বর্ন-এগেন শেল মানে ব্যাশ নয়। ব্যাশে বর্নের সমস্ত কলকব্জাই আছে, কিন্তু শুধু সেটুকুই নেই, আরো কিছু আছে। সেই আরো কিছুটা আমি কখনো কখনো ব্যাশ ম্যানুয়াল থেকে দেখেছি বটে, কিন্তু বিস্তর ফাঁক রয়ে গেছে। আর আপনারা নিজেরাই একটা করে বই লিখুন, বাংলায়, গ্নু-লিনাক্সের, সত্যি বলছি, এই বইটা লিখতে লিখতে গোটাটাই আমার মাথায় এত স্পষ্ট হয়ে উঠেছে, কোনো কিছু শেখার জন্যে একটা বই লেখার মত ভালো কোনো উপায় আর হয়না। আর এভাবেই তো বেঁচে থাকি আমরা। সব কিছুর পরেও। শূন্য নম্বর দিনের গোড়াতেই ছিল দুটো বাচ্চার কথা, দমদম স্টেশনে মেট্রোর কাউন্টারের পাশে বসে ভিক্ষে করত। কাল প্রায় সেই বয়সেরই দুটো বাচ্চাকে দেখলাম মধ্যমগ্রাম স্টেশনে। সাত আট বছরের দিদি দেড় দু বছরের ভাইকে কোলে নিয়ে দু-নম্বর প্ল্যাটফর্ম থেকে নামল লাইনে, তারপর লাইন পেরিয়ে এক নম্বরে আসছে। প্রতিটি বার পা ফেলার পর ওর হাঁটু বেঁকে যাচ্ছে, মুখ কুঁকড়ে যাচ্ছে, পাথরে এতটাই লাগছে, কিন্তু তাতে

দায়িত্বপালনে কোনো খামতি ঘটছে না। আমরা পোস্টমডার্নিজমে মানব জাতির ইতিহাসকে ব্যক্তিমানবের অতিক্রম করে যাওয়ার গল্প বলি, তার এর চেয়ে বড় উদাহরণ তো আর হয় না। চারদিকে যখন এ ওরটা কেড়ে খাওয়া, অন্যের খিদের থেকে চোখ ফিরিয়ে রাখাটাই নিয়তি ভবিতব্য এবং সামাজিক কানুন হয়ে দাঁড়াচ্ছে, তখনো তো প্যাঁচা জাগে, পাথরের পথে বাচ্চাতরকে কোলে বাচ্চা দাঁতমুখ খিঁচিয়ে পথ হাঁটে। তাহলে আমরা পারিনা কেন? নিজেদের মতো করে নিজেদের রেজিস্ট্যান্সের ব্যাকরণ আসুন নিজেরাই বানাই। অনেক কিছু দিয়ে তা ঘটে। এর কোনো বাঁধাধরা নিয়ম নেই। আসুন গ্নু-লিনাক্স দিয়েও আমরা খিদে জিইয়ে রাখার রাজনীতির বিরুদ্ধে এক এক পিস প্রতিরোধ বানাই। আমাদের তো রামও নেই, যে পিঠে রামের পাঁচ-আঙুলের সমাদরের দাগ নিয়ে গাছের ডালে ডালে লেজ বেঁকিয়ে লাফিয়ে বেড়াব, না-থাকুক, সেতুবন্ধের কাজে নিজের নিজের ব্যক্তিগত বাদামের খোলা তো থাকতে পারে।

যাকগে, যা বলছিলাম, ব্যাশের নিজেরই বরাদ্দ করা বিশেষ কিছু ভ্যারিয়েবল। 'PATH' বা 'PS1' ইত্যাদি এনভায়রনমেন্ট ভ্যারিয়েবলকে ধারণ করার ব্যাশপ্রক্রিয়ার কথা তো আমরা আগেই বলেছি। এর সঙ্গে ব্যক্তিগত লোকাল ভ্যারিয়েবল বানানো এবং ব্যবহারের কায়দা নিয়েও কথা বলেছি আমরা। এবার আমরা আগে থেকেই নির্দিষ্ট কিছু বিশেষ ব্যাশ ভ্যারিয়েবলকে দেখব। অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটারদের পরেই এদের কারণটা এখুনি দেখতে পাবেন। অঙ্কের সঙ্গে যাদের পরিচয়টা একটু কম, তাদের জন্যে একটু ভ্যারিয়েবল আর প্যারামিটার ধারণাদুটো নাড়িয়ে নিই। ধরুন একটা বাচ্চার রেজাল্ট নিয়ে ভাবছেন আপনি, তাকে বারবার বলছেন, পড়ার সময় বাড়া, সিনেমা দেখিস না, আড্ডা একটু কম মার। ইত্যাদি। কিন্তু একবারো বলছেন না, তোর বাবার আয়টা একটু বাড়া, তোর প্রতি তোর বাবা-মার ভালোবাসাটা একটু বাড়া, তোর চারপাশের বায়ুমণ্ডলে দূষণটা একটু কমা যাতে তোর হাঁফানিটা একটু কম হয়। কিন্তু নিজেই ভেবে দেখুন, এর সবগুলোই তো সমান রকমের গুরুত্বপূর্ণ। এর কারণ, এই মুহূর্তে আপনি চেয়ে চেয়ে মরে গেলেও, বা তার চেয়েও সাংঘাতিক, বিপ্লবী হয়ে গেলেও, এর কিছু কিছু জিনিষ আপনি কিছুতেই বদলাতে পারবেন না। আপনার সীমাবদ্ধ অধিকারের ভূমিতে, বাচ্চাটার লেখাপড়ার ভালো করার জন্যে মাত্র কয়েকটা জিনিষকেই আপনি বদলাতে পারেন। এগুলোই ভ্যারিয়েবল। আর যারা অপরিবর্তনীয় থাকছে, স্থির রয়ে যাচ্ছে, তারাই প্যারামিটার। আবার সিচুয়েশন থেকে সিচুয়েশনে এই ভূমিকাগুলোও বদলায়। ধরুন, আপনি সোজিওলজি ফিল্ডে লোক ঠকান, যেমন আমার পকেটমারার ব্যক্তিগত এলাকাটা হল অর্থনীতি, এবার আপনি একটা স্টাডি করছেন, সেখানে ব্যক্তি বাচ্চাটা আর নেই, আছে পরপর ইনকাম গ্রুপ, আর তাদের ঘরের বাচ্চাদের রেজাল্ট। এখানে রেজাল্ট নামক ফাংশনের ভ্যারিয়েবল হয়ে দাঁড়াল বাচ্চার মা-বাবার আয়, বাচ্চাটার ব্যক্তিগত পছন্দ অপছন্দ তথা আর সমস্ত কিছু এখন প্যারামিটার বা নিয়ন্তা। অন্য আরো প্রচুর উপাদানের সঙ্গে মিলে একটা অনড় স্থির গড় ব্যাপার হয়ে আছে প্রতিটি ইনকাম গ্রুপেই। ইকনমিক্সে এই ধরনের জন্যে সাইডলাইনের বাইরে বসে থাকা উপাদানগুলোর জন্যে একটা কাল্টনাম আছে — সিটেরিস পেরিবাস — অল আদার থিংস রিমেইনিং কনস্টান্ট — আর সমস্ত কিছু স্থির আছে এই অবস্থায়। ভারি নিরাপদ। যা যা বিষয় আপনার জন্যে অস্বস্তিকর, তাদের ঠেলে দিতে পারবেন এই সিটেরিস পেরিবাসে। এই এলাকার প্রতিটি উপাদানই তখন প্যারামিটার। যেমন, এই বাচ্চাটার উদাহরণে রেজাল্ট নামক ফাংশনের ভ্যারিয়েবল হল বাচ্চাটার পড়ার ইচ্ছে, দেখা সিনেমার সংখ্যা, আড্ডা মারার মোট সময়ের পরিমাণ। বাবার আয়টা তখন ছিল প্যারামিটার।

মানে, একটা বিশেষ পরিস্থিতিতে আপাত অপরিবর্তনীয় মালটা হল প্যারামিটার, আর বদলাতে থাকা বস্তুটা হল ভ্যারিয়েবল। এবার আমরা যে তালিকাটা আনব, তাতে দেখুন, একটা করে বিশেষ চিহ্ন বা চিহ্ন-সমাহার, আর তার বিশেষ একটা অর্থ। এই অর্থটা অপরিবর্তনীয়। সেই অর্থে তাই এদের নিয়ন্তা বা প্যারামিটার বলাই ভালো। কিন্তু ভ্যারিয়েবল নামটা চলে যখন, তার কোনো একটা আলাদা অর্থ বা ইতিহাস আছে নিশ্চয়ই, যেটা আমি জানিনা। ব্যাশ ম্যানুয়ালে দেখেছি, এদের স্পেশাল প্যারামিটার বলেই ডাকছে, ভ্যারিয়েবল নয়, আমাদের সাধারণ বুদ্ধিও তাই বলছে, এমনও হতে পারে যে প্রাটা ভুল করে এদের স্পেশাল ভ্যারিয়েবল বলে ডেকেছেন।

$# — 'set' কমান্ড দিয়ে আমরা যখন অবস্থানগত নিয়ন্তাদের মানগুলো দিয়ে দিচ্ছি ব্যাশকে, তখন আমাদের সেই আদেশে পোজিশনাল প্যারামিটারের মোট সংখ্যা। যেমন, একটু আগে, উট বলে ডাকার আমাদের উদাহরণটার সময়, আমরা যদি ব্যাশ প্রম্পটে '`echo $#`' কমান্ড দিতাম, ব্যাশ আমাদের দেখাত, '12'। গুনে

দেখুন, কেন এই সংখ্যাটাই দেখাচ্ছে ব্যাশ। তবে খেয়াল রাখবেন, পরের বার 'set' কমান্ডটা দেওয়ার আগে অব্দি বা লগ-আউট করার আগে অব্দি এটা লাগু থাকবে।

$* বা $@ — এরা দুটোই দেখায়, 'set' কমান্ড দিয়ে মোট যত অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটার ব্যাশকে দেওয়া হয়েছে, তার তালিকা। দুটোর মধ্যে একটাই তফাত আসে, যখন কোট চিহ্নের মধ্যে ব্যবহার করা হয় এদের। যেমন আমরা ব্যাশে লগ-ইন করছি যখন, '~/.bashrc' ফাইলে ওই 'set $(date)' কমান্ডের কারণে 'echo $*' বা 'echo $@' করলেই, লগ-ইনের মুহূর্তে 'date' কমান্ড যা দেখিয়েছিল, পোজিশনাল প্যারামিটারের জমিতে তখন ভরে রাখা সেই গোটাটা ব্যাশ দেখিয়ে দেবে আমাদের। করে দেখুন। বা উটের উপাখ্যানে, আরব কাফেলার বিষয়ে গোটা ওই শব্দবন্ধটা। কিন্তু প্রতিবার 'set' কমান্ড দেওয়া মাত্রই আগেরটা মুছে নতুন মান ভরে দেওয়া হবে পোজিশনাল প্যারামিটারের জমিতে।

$! — 'echo $!' কমান্ড দিলে ব্যাশ আপনাকে দেখায় শেষ যে প্রক্রিয়া বা প্রসেসটা ব্যাকগ্রাউন্ডে গেছে, তার পিআইডি (PID) বা প্রক্রিয়া-সূচক। ব্যাকগ্রাউন্ডের আলোচনা আজকেরই ৪ নম্বর সেকশনে আমরা করে এসেছি। ধরুন আপনি একটা ইন্টারাক্টিভ শেল প্রক্রিয়া চালু করলেন, কিন্তু তাকে রাখলেন ব্যাকগ্রাউন্ডে। 'bash &' কমান্ড দিয়ে। এই অ্যাম্পারস্যান্ড বা '&' চিহ্নটা তো আমরা চিনি, একটা প্রক্রিয়াকে সক্রিয় সম্মুখভূমি থেকে আপাতনিষ্ক্রিয় পশ্চাৎভূমিতে ঠেলে দেয়। যেই আপনি এটা করলেন স্ক্রিনে আপনাকে ব্যাশ এই জাতীয় কিছু একটা দেখাবে — '[1] 1301'। এর অর্থটা মনে করতে পারছেন, প্রথম চৌকো ব্রাকেটের মধ্যে '[1]' বস্তুটা হল জব নাম্বার বা বকেয়া কাজের সূচক, আর পরের '1301' হল এর পিআইডি। এবার আপনি যদি 'echo $!' কমান্ড দিয়ে এই '$!' প্যারামিটারটা জানতে চান, ব্যাশ আপনাকে ওই পিআইডিটা দেখাবে, মানে, '1301'।

$$ — 'echo $$' আপনাকে দেখায় এই মুহূর্তে চালু ব্যাশ শেলের পিআইডি। একদম হাতে নাতে পরখ করে নিন। এই মাত্র আপনি একটা ব্যাশকে ব্যাকগ্রাউন্ডে পাঠিয়েছেন। একে ফোরগ্রাউন্ডে আনতে তো আমরা জানি। জাস্ট 'fg' করুন। বা, আরো কাউকে কাউকে আপনি যদি পশ্চাৎপদ এলাকায় পানিশমেন্ট পোস্টিং করে থাকেন, তাহলে তার ব্যাকগ্রাউন্ড ব্যাশের জব নম্বর দিয়ে করুন। ব্যাশ চালু হয়ে গেল। এখন আপনি আছেন আমাদের সেই চেনা ঘাপলা ব্যাশের পেটে ব্যাশে। তাতে অবশ্য ফারাক কিছু হবেনা ব্যাশের আচার-আচরণে। এবার কমান্ড দিন 'echo $$'। দেখুন ব্যাশ আপনাকে দেখিয়ে দিল, সেই '1301', আপনি তো চেনেন, ব্যাকগ্রাউন্ডে পাঠানোর সময় এইমাত্র ব্যাশ আপনাকে দেখিয়ে দিয়েছিল ব্যাশের পেটে ব্যাশের ওই নতুন প্রক্রিয়াটার পিআইডি।

$0 — এই মুহূর্তে ব্যাশ যে কমান্ডটা পালন করছে, তার নাম ভরা থাকে এই বিশেষ প্যারামিটারে। এটার একদম হাতেগরম একটা প্রয়োগ হতে পারে, আমাদের একটু আগের ৯ নম্বর সেকশনের 'writefiles' নামের ব্যাশস্ক্রিপ্টটাই। ধরুন, একটু একটু বদলে আপনি ওই একই রকম কাছাকাছি কাজের বেশ কয়েকটা আরো স্ক্রিপ্ট বানিয়েছেন, 'writefiles1', 'writefiles2' ইত্যাদি নামে। এবার, 'writefiles' স্ক্রিপ্টটার একদম শেষ লাইনে, মানে, 'done' লাইনটায় লুপ শেষ করার পর, যদি আর একটা ঠিক এই রকম লাইন যোগ করেন, 'echo -e "\n\n$d\n[Made by $0 on $(date)]">>$f', তাহলে কী হবে বলুন তো? বেশ একটা মেড-ইন-ইন্ডিয়া গোছের লাগছে না? এখানে আমি ইচ্ছে করে এটা বোঝালাম না, ওখানের আলোচনার সঙ্গে মিলিয়ে পড়লে নিজেই বুঝতে পারবেন। ফলাফলটা হবে এই যে, এইরকম একটা অংশ 'writefiles' দিয়ে বানানো ফাইলের শেষে যোগ হয়ে যাবে —

```
**********************************************
[Made by ~/bin/writefile on Sat Mar 20 05:16:22 IST 2004]
```

$? — এটা শেষ ব্যাবহৃত ব্যাশ কমান্ডের এক্সিট স্ট্যাটাস বা নিষ্ক্রমণ স্থিতি দেখায়। মানে, কমান্ডটা তার কাজ করতে পেরেছে কি পারেনি। এদিকে ব্যাশ বেশ নাগার্জুনপন্থী, মানে তার কাছে সাফল্য হল শূন্য। যদি 'echo $?' করলে '0' দেখায়, তার মানে কাজটা হয়েছে। সি-তেও কিন্তু তাই।

$- — ব্যাশ শেলের অপশান। 'echo $-' কমান্ডটা দেখায়, যে ব্যাশের ভিতরে দাঁড়িয়ে আপনি কমান্ডটা দিলেন সেটা কী কী অপশান নিয়ে চালানো হয়েছে। ম্যান পড়ে দেখুন, গাদা গাদা অপশান আছে।

১৬।। গ্লব এক্সপ্রেশন

অতিচিহ্ন বা মেটাক্যারেকটার এইসব চমকেবল রাশভারি নাম দিয়ে যাদের বোঝানো হচ্ছে সেইসব ভ্যাবলাকান্ত চিহ্নগুলোকে আমরা ব্যবহার করে আসছি সেই গোড়া থেকেই, এখানে একটা ওখানে একটা এই ভাবে ব্যবহার করে এসেছি। '>', '|', '\', '&', '&&', '*', '.' ইত্যাদি। এবার তাদের একটু গুছিয়ে চিনব। এদের মধ্যে একটা গুরুত্বপূর্ণ অংশ হল পাইকিরি ছক বা গ্লব এক্সপ্রেশন। একটা একটা করে চুন-চুনকে মারুন্গার গল্পে যখন আর বিশ্বাস করা যায়না, গণহত্যাপন্থী হয়ে ওঠার প্রয়োজন পড়ে, তার জন্যেও ব্যবস্থা করা আছে আপনার সাধের গ্নু-লিনাক্সের সাধের ব্যাশে। অন্য আর এক নামেও এদের বোঝানো হয়, আগেই বলেছি, রেগেক্সপ বা রেগুলার এক্সপ্রেশন (RegExp)। খুচরো নড়বড়ে ব্যবহারকারীর কাঁপা কাঁপা ভিতু রকমে একটা একটা করে ফাইল খোঁজা নয়, এক লপ্তে লাটকে লাট ফাইল পাইকিরি রকমে নাড়াচাড়া করার জন্যে লাগে এই পাইকিরি ছক বা গ্লব এক্সপ্রেশন।

ভালো করে বোঝার জন্যে প্রথমে একটা এলেবেলে ডিরেক্টরি বানিয়ে নিন, তাতে ফাইল রাখুন নিচের তালিকামত। সব ফাইলই হল মন্ত্রী-উবাচের মত, মানে ফাঁপা এবং ফাঁকা। এরকম ফাইল বানিয়ে নিন লাটে। আপনি টাচউড, আগেই বলেছি, টাচ করুন আর ফাইল পয়দা হয়ে যাবে। এরকম নামগুলো ইচ্ছে করেই দিচ্ছি আমরা, যাতে পাইকিরি ছকের ব্যাকরণটা মালুম করা যায়। আপনি বিশ্বাস করবেন কিনা জানিনা, আমি আমার ফাইলের এত খারাপ নাম দিই-না, এত খারাপ ফাইলের নাম দেওয়া যায়না, কিন্তু এখানে এরকম নাম দেওয়ার কিছু কারণ আছে, এখুনি দেখতে পাবেন। সেই কারণগুলোকে মিলিয়েই হয়ত এর চেয়ে কম কদর্য নাম দেওয়া যেত, কিন্তু আমার মুন্ডুর মোট রসদ গত পাঁচমাস এই লেখায় রয়েছে, কল্পনাশক্তি ইত্যাদি অন্য প্রসেসগুলো সব নতুন পড়া গরমে জিভ বার করে ধুঁকছে। এখানে ফাইল মোট চবিবশটা। প্রবন্ধ ষোলটা, ইংরিজির তিন, ভূগোলের চার, ইতিহাসের তিন আর অঙ্কের ছয়খানা। চারটে বিষয়ের চারটে বিবলিওগ্রাফি এবং চারটে নোটস। তাও তো, শুধু ছোট হাতের নাম দিয়েছি, বড় হাতের অক্ষর ব্যবহারই করিনি, জাটিল্য তাতে আরো বাড়ত।

essay.english.1.text	essay.geo.4.text	essay.math.3.text	essay.history.biblio
essay.english.2.text	essay.hist.1.text	essay.math.4.text	essay.math.biblio
essay.english.3.text	essay.hist.2.text	essay.math.5.text	notes.english.text
essay.geo.1.text	essay.hist.3.text	essay.math.6.text	notes.geography.text
essay.geo.2.text	essay.math.1.text	essay.english.biblio	notes.history.text
essay.geo.3.text	essay.math.2.text	essay.geography.biblio	notes.math.text

ধরুন এটা করছি হোম ডিরেক্টরির মধ্যে 'essay' বলে একটা সাবডিরেক্টরিতে। এবার এই '~/essay' ডিরেক্টরিতে দাঁড়িয়ে, '`ls`' মারলে, এই ফাইলগুলোই দেখতে পাব আমরা। বা '`ls *`' মারলেও তাই হবে, আমরা জানি, '*' চিহ্নটার মানে সব কিছু সমস্ত কিছু। শুধু ডটানন ফাইলদের ধরতে পারেনা চিহ্নটা, মনে পড়ছে, ছয় নম্বর দিনের আমাদের আলোচনা? কিন্তু যদি আমরা কমান্ড দিই, '`ls essay.[gh]*.text`', তাহলে আমরা এই ব্রিলিয়ান্ট নামের ফাইলগুলোর গোটা তালিকাটা আর দেখতে পাব না, হায়, দেখব শুধু ভূগোল আর ইতিহাস, 'geo' আর 'hist' সংক্রান্ত প্রবন্ধাবলীর তালিকা। ভূগোলের চার আর ইতিহাসের তিন পিস। এই কাঠামোটাই একটা পাইকিরি ছক বা গ্লব এক্সপ্রেশন।

essay.geo.1.text	essay.hist.1.text
essay.geo.2.text	essay.hist.2.text
essay.geo.3.text	essay.hist.3.text
essay.geo.4.text	

মানে, এই ছকটাকে ব্যাশ কী ভাবে বুঝছে সেই গোটাটা ভাবুন। প্রথমে মেলাচ্ছে সেই সমস্ত ফাইল যাদের শুরুতেই আছে পাইকিরি ছকের গোড়ার অংশটা, '`essay.`'। বিন্দুটাকে দেখতে আমি কখনোই ভুল করিনা, প্রথমত, আমার মা-র ডাকনাম পুঁটকি, আর দ্বিতীয়ত, বিন্দুতে যে কী বীভৎস সব কেলো ঘটে যায় তা ইউক্লিডের শ্বশুরও তেমন জানতেন না, গ্নু-লিনাক্স করতে গিয়ে আপনি যেমন জানবেন। তার মানে, ব্যাশ তুলে নিচ্ছে মোট কুড়িটা ফাইল, এবং বাদ দিচ্ছে চারটে, কারণ তারা '`essay.`' দিয়ে শুরু নয়। তারপরেই ব্যাশ যাচ্ছে ছকের দ্বিতীয় অংশটায়, দেখছে চৌকো ব্রাকেটের ভিতরকার অক্ষরগুলো। তুলে নেওয়া কুড়িটা ফাইলের মধ্যে যারা '`g`' বা '`h`' দিয়ে শুরু সবাইকে সে

তুলে নিচ্ছে তার হোলসেল ঝোলায়। তার মানে এই ধাপে বাদ পড়ে যাচ্ছে ইংরিজির তিনটে প্রবন্ধ একটা বিবলিওগ্রাফি মোট চারটে, এবং অঙ্কের মোট সাতটা ফাইল, কারণ তারা 'g' বা 'h' দিয়ে শুরু নয়। মানে, মোট এগারোটা ফাইল বাদ যাচ্ছে। কুড়ি থেকে এগারো গেল, পড়ে রইল নয়। কিন্তু কাজ এখনো শেষ হয়নি। এরপরে এই তালিকা থেকে সেই ফাইলগুলো ব্যাশ বাদ দিয়ে দিচ্ছে যাদের শেষে আবার সবিন্দু ওই '.text' অংশটা নেই। এই ধাপে দেখুন বাদ পড়ছে দুটো ফাইল, 'essay.geography.biblio' এবং 'essay.history.biblio'। কারণ, এদের শেষে পাইকিরি ছকের শেষ অংশটা, '.text', নেই। তার মানে নটা থেকে দুটো বাদ গেলে পড়ে থাকছে ওই সাতটা ফাইল, এইমাত্র টেবিল করে যাদের আমরা দেখালাম। এবার বলুন তো আগের পাইকিরি ছক বা গ্লব এক্সপ্রেশনটা স্লাইট একটু বদলে যদি এরকম কমান্ড দিই, 'ls essay.[^gh]*.text', এতে ব্যাশ কোন ফাইলগুলোকে দেখাবে? এখানে এই '^' চিহ্নটা ব্যাশকে বলে দিচ্ছে শুধু সেই অক্ষরেরা যারা 'g' বা 'h' নয়। ফাইলের তালিকাটা আমি দিয়ে দিচ্ছি, বুঝে নিন আপনি নিজে, এইমাত্র করা আলোচনাটার সঙ্গে মিলিয়ে। তফাত শুধু একটাই, 'g' বা 'h' এই সমাহারটার বদলে সেই সমাহার যা তৈরি তাদের নিয়ে যারা 'g'-নয় এবং 'h'-নয়।

essay.english.1.text	essay.math.1.text
essay.english.2.text	essay.math.2.text
essay.english.3.text	essay.math.3.text
	essay.math.4.text
	essay.math.5.text
	essay.math.6.text

এখানে যেমন আমরা চৌকো ব্রাকেটের মধ্যে সেই সব অক্ষরগুলো দিয়ে দিলাম যাদের দিয়ে শুরু করাতে চাই আমরা ব্যাশকে, সেরকম যদি অনেকগুলো অক্ষরগুলো হত, তাহলে সেটাকে একটা ধারাবাহিকতা হিশেবেও দিতে পারতাম। ধরুন, আমরা শুধু ইংরিজির ভূগোলের আর ইতিহাসের প্রবন্ধ কটাই দেখতে চাইছি, এই দশটা ফাইল, তখন আমরা চাইব শুধু 'essay.[e-h]*.text' ফাইলগুলোকে। এর মানে মধ্যের অংশটায় 'e' থেকে 'h' অব্দি যে কোনো বর্ণ দিয়ে শুরু থাকলেই তাকে অন্তর্ভুক্ত করবে এই ছকটা। এটা অন্য আর এক ভাবেও করা যেত, পাইকিরি ছক যদি দেন 'essay.{english,geo,hist}.text' সেটাও ধরবে সেই সমস্ত প্রবন্ধদের যাদের মধ্যের অংশটায় আছে, 'english' বা 'geo' বা 'hist'। এই সংক্রান্ত আরো কোটি কোটি ডিটেইলস আছে, তার অনেকগুলোই ভারি মজার। ম্যান পড়ার নেশা করো ম্যান ... ইত্যাদি।

শুধু অঙ্কের প্রবন্ধের ফাইল কটাকে চাই যদি, আমার পাইকিরি ছকটা হবে, 'essay.math.[1-6].text'। এর বদলে, যদি আমরা পাইকিরি বাজার করার জন্যে ছক দিতাম, 'essay.math.?.text' ছকও দিতে পারতাম। তাতেও একই কাজ করত। কিন্তু '?' যেহেতু একটা কোনো বর্ণ বা অঙ্ককে বোঝায়, এখানে যদি দু-অঙ্কের কোনো সংখ্যা হয়, তাকে আর তুলবে না ব্যাশ। ধরুন, আপনার এলাকায় আপনি একটা জনগনতান্ত্রিক অঙ্কবিপ্লবের বিপ্লবী ভ্যানগার্ডদের দলে আছেন, এবং অঙ্কের প্রবন্ধ লিখেই চলেছেন, ঘ্যাশাঘ্যাশ ঘ্যাশাঘ্যাশ, শেষতমটা চলছে 'essay.math.96.text'। এই অবস্থায় গোটাটাকে আনতে হবে দু-অঙ্কের জায়গা বানিয়ে, মানে 'essay.math.??.text'। তার চেয়ে ভালো হয় যদি পাইকিরি ছকটা দেন 'essay.math.*.text', এতে করে আপনি হাজার লাখ কোটি প্রবন্ধ ছড়াতে থাকলেও ব্যাশের কোনো অসুবিধে হবেনা। কিন্তু সাবধান, যদি 'essay.math.*' দিয়ে ছেড়ে দেন, তাতে 'essay.math.' দিয়ে শুরু সব ফাইলকেই টেনে আনবে ব্যাশ। মানে অঙ্কের বিবলিওগ্রাফি নোটস সবকিছুই। এরকম আরো অজস্র ছক হয়, দুনিয়াটা তো নিয়ন্ত্রণ করছেই পাইকাররা, আমরা খুচরোরা নিতান্তই খুচরো। খুচরো মানুষের ইগো-ক্রাইসিস থেকে বাঁচতে গ্নু-লিনাক্স ব্যাশের গ্লব ব্যবহারের অভ্যেশ করার প্রেসক্রিপশন খুবই হচ্ছে আজকাল, এতে কাউন্সেলিং করানোর খরচ কমে আসছে।

এখানে কমান্ড লাইনে পাইকিরি ছক ব্যবহারের বিষয়ে একটা কথা বলে নিই। ধরুন আপনি একটা অঙ্কের প্রবন্ধের নাম যদি বদলাতে চান, 'mv essay.math.1.text math.1', ব্যাশ বিনা আপত্তিতে আপনার অঙ্কের প্রবন্ধটার সামনে আর পিছন থেকে 'essay' আর 'text' অংশদুটো উড়িয়ে দেবে। কিন্তু এবার এই কাজটা লাটে করতে যান, পাইকিরি ছকে, 'mv essay.math.*.text math.*', দিয়ে দেখুন, ব্যাশ রাজি হবেনা, অস্বীকার করবে, প্রমাদ বার্তা পাঠাবে স্ক্রিনে। কিন্তু কপি করা যাবে দিব্য। আপনি একটা ডিরেক্টরি বানান, 'mkdir math' দিয়ে।

এবার সেখানে সব কটা অঙ্কের প্রবন্ধ কপি করে দিন, 'cp essay.math.*.text math/', দেখুন, চোখের পলক না ফেলতেই হয়ে গেল, কোনো আপত্তিই করল না ব্যাশ। 'ls math/' মেরে আপনি দেখেও নিলেন। এ কেমন ধাঁচের বিদঘুটেপনা? কেন একটা করছে, অন্যটা করছে না? আসলে 'mv' কমান্ডটার আভ্যন্তরীন কাঠামোটা খেয়াল করুন। কমান্ডটার দুটো অংশ থাকে। 'mv <name1> <name2>' হল তার আকার। আমরা একটা ফাইলকে নড়াই অন্য একটা ফাইলে বা ডিরেক্টরিতে। যাকে নাড়াচ্ছি তার নাম 'name1', আর যেখানে নাড়াচ্ছি, তার নাম 'name2'। যখন 'name2' হল একটা ডিরেক্টরির নাম, মানে, নড়িয়ে আমরা একটা ডিরেক্টরিতে রাখছি, সেটাই ভৌত অর্থে নাড়ানো। আর 'name2' যখন একটা ফাইলনাম, তার মানে আমরা নড়িয়ে অন্য ফাইলে, মানে অন্য নামের ফাইলে রাখছি, মানে ফাইলটার নাম বদলাচ্ছি। 'mv' কমান্ডটা এর বাইরে কোনো কাঠামোর আদেশ নিয়ে কাজ করতে পারছেনা। এবার মজার কথা, আমাদের এইমাত্র দেওয়া আপাতদৃষ্টিতে তো এই কাঠামোটা মেনে নিচ্ছে। তাহলে কেলোটা কোথায়? কেলোটা রয়েছে ব্যাশের ক্রিয়ার মধ্যে। আপনি যখন পাইকিরি ছক দিচ্ছেন, ব্যাশ সেটাকে বাড়িয়ে নিচ্ছে, এক্সপ্যান্ড করে নিচ্ছে। আপনার দেওয়া 'mv' কমান্ডের প্রথম অংশের পাইকিরি ছকটা বেড়ে গিয়ে হয়ে দাঁড়াচ্ছে ছ পিস খুচরো ফাইল। বুঝতে পারছেন, 'mv' কমান্ডের কাঠামোটাকেই আর খুঁজে পাচ্ছে না ব্যাশ। তাই কাজও করছে না। কিন্তু 'cp' কমান্ডের বেলায় সেরকম কোনো সমস্যা নেই, ছ খানা ফাইলের পরে আসছে তাদের পাঠানোর ঠিকানা, তাই ব্যাশের কাজ করতেও কোনো অসুবিধে হচ্ছে না।

কিন্তু ফাইলের নাম কি তাহলে লাটে বদলানো যাবেনা? ফাইল বদলানোর কাজটা খুচরো, নিতান্তই খুচরো। এই রে, বদলই তো জীবন, বেঁচে থাকা। আবার তো ইগো ক্রাইসিস? না, উপায় আছে, যদি বাড়াতে চান, সেটা ভারি সহজ, আগে নামগুলো ছোট করে নি, তারপর সেই ছোট হওয়া নামটা বাড়িয়ে দেখাচ্ছি। আগে ছোট করাটা দেখাই, সেটায় একটু মনোযোগ দিতে হবে। প্রথমে একটা ভ্যারিয়েবল বানান। আপনার একান্ত নিজস্ব একপিস লোকাল জিনিষ, যা আমরা কোটি কোটি বার করেছি।

```
filename=essay.math.text
```

একবার প্রতিধবনি মেরে দেখেও নিতে পারেন, মালটা কী দেখাচ্ছে। জানি ব্যাশ ভুল করেনা, তবু দেখে নিতে দোষ কী? আপনার তো দিতে কোনো ভুল হতে পারে। সেই পিলারের একটা এল-এর মত, তাতেও পিলার খাড়ায় ঠিকই, কিন্তু দুটো দেওয়াই ভালো, একটু পোক্ত হয়। এবার একটা কমান্ড দিন

```
echo ${name##essay.}
```

দেখুন, কমান্ড প্রম্পটে ফুটে উঠেছে, 'math.text'। মানে পাঁঠার মুণ্ডটা বলি করে দিয়েছে, গোড়ার 'essay.' অংশটা। কিন্তু আমার পূজ্যপাদ অভিভাবকবৃন্দ যে কাজে আমার দক্ষতা অনস্বীকার্য বলে স্বীকার করে নেন, সেইরকম যদি আর কেউ হয়, সেও তার নিজের পাঠা লেজে কাটতে চায়, '.text' অংশটা ওড়াতে চায়, তাহলে? আবার একটা কমান্ড দিন

```
echo ${name%%.text}
```

এবার দেখুন, কমান্ড প্রম্পটে ফুটে উঠেছে, 'essay.math'। এই ম্যাজিকের মন্ত্রটা বুঝে নিন আগে। দুই জবাইয়ের দুরকম জবাফুল। শুরু ছাঁটার জন্যে '##' আর শেষ ছাঁটার জন্যে '%%'। '##' ব্যাশকে বলছে শুরু থেকে মানে বাঁ-দিক থেকে এই 'essay.' স্ট্রিং বা চিহ্নমালাটা অব্দি গোটাটা ছেঁটে ফেলো, আর '%%' ব্যাশকে বলে দিচ্ছে, শেষ থেকে মানে ডান-দিক থেকে এই '.text' স্ট্রিং অব্দি গোটাটা ছেঁটে ফেলতে। ভালো করে যদি জটিলতাটা বুঝতে চান, তাহলে খেজুরগাছের কাছে যান, এতবার যেতে যেতে কাঁটাগুলোও অনেক মসৃণ হয়ে গেছে এতদিনে। এবার, প্রকৌশলটা যখনই শিখে গেলেন এটাকে লাটে প্রয়োগ করুন পাইকিরি ছকে। আর ডিরেক্টরি তো আমাদের তৈরি আছে। এইমাত্র আমরা 'math' বলে ডিরেক্টরিটা বানিয়ে সেখানে কপি করে নিয়েছি অঙ্কের সমস্ত প্রবন্ধ। 'math' ডিরেক্টরিতে ঢুকে 'ls' কমান্ড দিয়ে দেখুন, ব্যাশ আপনাকে যাবতীয় ফাইলের তালিকা তুলে দিচ্ছে।

```
essay.math.1.text  essay.math.3.text  essay.math.5.text
essay.math.2.text  essay.math.4.text  essay.math.6.text
```

মানে, এই ডিরেক্টরির ফাইলগুলোয় গোড়ায় একটা 'essay.' দাড়ি, আর শেষে একটা '.text' লেজ। প্রত্যেকটা রামছাগলের মত লম্বা ফাইলের দাড়ি আর লেজ ছেঁটে দিন। স্টেপ বাই স্টেপ এগোন। প্রথমে দাড়ি। পরে লেজ। কমান্ড দিন

```
for i in `ls *math*`; do mv $i ${i##essay.};done
```

এটা একটা 'for' লুপ। আগে একবার ধরে বোঝানো আছে, এখন দেখুন, প্রথমে ব্যাশ 'ls *math*' করে গোটা ডিরেক্টরিতে, মধ্যে 'math' অংশটা আছে এমন সব ফাইলের একটা তালিকা করে নিচ্ছে, এবার সেই তালিকাটাকে আমরা ডেকে নিয়েছি কমান্ডের মধ্যে কমান্ডে ব্যাককোট দিয়ে। এবার তার থেকে এক একটা করে ফাইল পড়ছে ব্যাশ এবং তাকে আমাদের এইমাত্র দেওয়া বলিপ্রক্রিয়ায় গোড়ার 'essay.' অংশটা ছেঁটে দিচ্ছে। এবার কমান্ড প্রম্পটে একবার '`ls`' মারুন। দেখুন ফাইল দেখাচ্ছে

```
math.1.text     math.2.text     math.3.text     math.4.text     math.5.text
math.6.text
```

মানে, দাড়িছাঁটার কাজ ফিনিশ। এবার লেজ। কমান্ড দিন

```
for i in `ls *math*`; do mv $i ${i##essay.};done
```

কমান্ড প্রম্পট ফেরত এলে আর একবার '`ls`' মারুন। এবার দেখুন, ছোট্ট ছোট্ট চিকণ ছাগশিশুর মত ফাইলে ভরে গেছে আপনার আদিকচক্রবাল ডিরেক্টরি। আহা তারুণ্যের কী স্নিগ্ধতা। কারুর নামে কোনো রেয়াজি মেদের কণাটুকু অব্দি নেই।

```
math.1  math.2  math.3  math.4  math.5  math.6
```

যা খাটলাম আপনাদের জন্যে যথেষ্ট। পাইকিরি ছক নিয়ে আর নয়। এবার আমরা ধরব অন্য মেটাক্যারেকটার বা অতিচিহ্ন। তার পর একটু লুপের দিকে আলগা লুক দেওয়া। তারপরেই ছুটি। ও, নাম ছোট করা তো হল, এবার এদের একবার বড় করে নেওয়া যাক। তার জন্যে কমান্ডটা খুবই সহজ, এবং একটা কমান্ডেই এই ছাগলছানাদের দাড়ি এবং লেজ সহ পরিপূর্ণতায় পৌঁছে দেওয়া যাবে।

```
for i in *; do mv $i essay.$i.text; done
```

এবার '`ls`' মারলেই দেখতে পাবেন, ফাইলগুলো আবার তাদের যাত্রাশুরুর এক নম্বর চৌখুপিতে পৌঁছে গেছে। এই যে বারবার '`do`' আর '`done`' দিয়ে করছি, এটা কিন্তু আদৌ জরুরি নয়, আসলে লুপের স্ক্রিপ্টোচিত কাঠামোর সঙ্গে চেনাশোনাটা বাড়িয়ে রাখতে চাইছি। আর একটা ছোট উদাহরণ দিয়ে রাখি এখানে, যা স্ক্রিপ্ট বানাতে গিয়ে বেশ কাজে লাগে। পথ-সহ একটা ফাইলনামের গোটাটার থেকে তার নিছক ফাইলনাম আর পথ — এই দুটোকে আলাদা করে বার করে নিতে পারে ব্যাশ, 'dirname' আর 'basename' কমান্ড দুটো দিয়ে। ধরুন, পথ বা ঠিকানা সহ একটা ফাইল, '/usr/local/mplayer/conf'। এটা দেখে আমরা সচেতন মানুষেরা নিজে নিজেই বুঝে নিতে পারি, '/usr/local/mplayer' হল ফাইলটার ঠিকানা বা পথ মানে 'dirname', আর 'conf' হল নিছক ফাইলনাম মানে 'basename'। এবার, '`dirname /usr/local/mplayer/conf`' কমান্ড দিলে, ব্যাশ তার কমান্ড প্রম্পটে দেখাবে, '`/usr/local/mplayer`'। আর '`basename /usr/local/mplayer/conf`' কমান্ড দিলে দেখাবে '`conf`'। এই দুটোর সুন্দর ব্যবহার আছে অনেক, কিন্তু প্রথমত সেগুলো অনেক পরিণত স্ক্রিপ্টে, আর সহজ কোনো উদাহরণ মাথায় ভেবে নিতে পারছি না। তার খুব একটা দরকারও নেই, এই অর্থে যে, আজকের এই গোটা আলোচনাটা আপনাকে এগিয়ে দিতে চাইছে স্ক্রিপ্ট লেখার আগেও স্ক্রিপ্ট পড়ার দিকে। সিস্টেমের মধ্যেই যে অজস্র স্ক্রিপ্ট এমনিতেই আছে, তারা কোথায় কখন ঠিক কী করে — সেই বোঝাটার জন্যেই দরকার পড়ে স্ক্রিপ্ট পড়া। আর এই স্ক্রিপ্টগুলো তাদের লেখা যারা সবচেয়ে ভালো করে এই কাজটা জানে। তাই, শেখার জন্যে তারাই আদর্শ। আজকের আলোচনাটা জাস্ট সেই শেখাটার দিকে পৌঁছে দেওয়া। এখন থেকে যাতে, কোনো স্ক্রিপ্ট সামনে দেখে আপনি আন্দাজ করার চেষ্টা শুরু করতে পারেন, অন্তত কোথায় কোথায় বুঝতে পারছেন না সেইটুকু বুঝতে পারেন। একটা নতুন জিনিষের বেলায় সেই শেখাটাই কিন্তু একটা বিরাট ব্যাপার। এই পাইকিরি ছকের কাজ শেষ। এবার এগোনো যাক অন্যান্য অতিচিহ্নের দিকে।

১৭।। কয়েকটা মেটাক্যারেকটার বা অতিচিহ্ন

এই অতিচিহ্ন বা মেটাক্যারেকটারদের এই নামে ডাকার কারণ এই যে, এরা নিজেরা নিজেরা যেমন নিছক একটা চিহ্ন, তেমনি ব্যাশের কাছে এদের বিশেষ কিছু অর্থও আছে। কোন অবস্থায় এদের চিহ্ন হিশেবেই পড়বে ব্যাশ, কোন অবস্থায় সেই বিশেষ অর্থে তারও নিয়মকানুন আছে। সেটা আমরা এখুনি জানব।

শব্দযতি বা ওয়ার্ড সেপারেটর

সচরাচর একে আইএফএস বলে ডাকা হয় (IFS — **I**nternal-**F**ield-**S**eparator)। আমরা 'set' কমান্ড দিয়ে ব্যাশের এনভায়রনমেন্ট ভ্যারিয়েবলদের পড়তে শুরু করেছি সেই ছয় নম্বর দিন থেকেই। এবার সেই 'set' কমান্ডের ফলাফলকে একটু গ্রেপ মারা যাক। `set| grep 'IFS'`। এতে যে ফলাফল আপনার স্ক্রিনে দেখাবেন ব্যাশবাবু, তাতে এজাতীয় একটা লাইন পাবেনই, `IFS=$' \t\n'`। এই লাইনটার ডানদিকের অংশটা খেয়াল করুন। দুটো কোটের মধ্যে রয়েছে তিনটে জিনিষ। একটা ফাঁকা ভূমি। তারপর `\t` মানে ট্যাব চিহ্ন। তারপর একটা নিউলাইন `\n`। শূন্য নম্বর দিনে এবং তার পরেও অ্যাসকি কোড আলোচনার সূত্রে আমরা ছুঁয়ে এসেছি এগুলো, মনে করুন। এই তিনটে জিনিষের যে কোনোটাকেই ফিল্ড সেপারেটর বলে ধরে নেয় ব্যাশ। একটু আগে আমরা যখন 'set' কমান্ড দিয়ে অবস্থানগত নিয়ন্তাদের বানাচ্ছিলাম, মনে আছে, এক একটা করে স্পেস পেয়েছে ব্যাশ, আর এক একটা করে নতুন প্যারামিটার বানিয়ে নিয়েছে। ওখানে ট্যাব বা নিউলাইন হলেও একই হত। নিউলাইন মানে কিন্তু ক্যারেজ রিটার্ন নয়। ক্যারেজ রিটার্ন পাই আমরা এন্টার চাবি টিপে। মানে, এই আমার কমান্ড খতম, এবার বাছা ব্যাশ তুমি পিন্ডি চটকাতে শুরু করো। এই নিউলাইন বা `\n` তৈরি হয় যখন আমরা একটা `\` দিয়ে তারপর এন্টার মারি। এই ভাবে আমরা খুব লম্বা কমান্ডকে এক লাইন থেকে আর একটা লাইনে টেনে নিয়ে যেতে পারি, তাতে কমান্ড সমাপ্ত হয়না। এমনিতেও ব্যাশ স্ক্রিনের চওড়ায় না ধরলে স্ক্রিনে পরের লাইনে নিয়ে যায়, কিন্তু তা আসলে স্ক্রিনের নতুন লাইন, কিন্তু ব্যাশের কাছে নয়, উপরের লাইনটা শেষ হয়ে যেতে পারে যে কোনো শব্দের মাঝখানে, এই `\` দিয়ে তারপর এন্টার মেরে আপনি আপনার পছন্দমত জায়গায় ভাঙতে পারেন কমান্ডটাকে। শব্দযতি বা ওয়ার্ড সেপারেটর নামটার চেয়েও ফিল্ড সেপারেটর বা জমির আল বললে ব্যাপারটা আরো ভালো বোঝা যায়। দুটো জমির মধ্যে সীমাটা স্লিম হতে পারে, একটা স্পেসের, ট্যাবমার্কের মত আদনান সামি সাইজের হতে পারে, আবার দুটো জমির মধ্যে বয়ে যাওয়া নিউলাইনের নদীও তো জমিদুটোকে বিশ্বস্ত ভাবেই আলাদা করে।

সারিবদ্ধ আদেশের সেমিকোলন `;`

ইনিও আমাদের চেনা। আমার গদ্য শেখার শুরু যার হাতে, সেই দান্তে, ভাষ্করজ্যোতি সমাস করেছিল, দাঁত-অন্তে-যাহার, বহুব্রীহি, ছড়া চলত, দান্তের গাট্টা টাকায় আটটা, শিশিরবাবু স্যার বলেছিলেন, বাক্যটা শেষ করলেই করতে পারতিস, কিন্তু করলি না, একসাথে আরো কিছু বলে নিতে চাইছিস — এর মানে সেমিকোলন। ঠিক তাই ব্যাশের বেলাতেও। আলাদা ভাবে একটা কমান্ড দেওয়ার পরে এন্টার মেরে ব্যাশকে দিয়ে কাজ করিয়ে নিতে পারতাম, কিন্তু একসঙ্গে দিয়ে দিলাম একাধিক কমান্ড, এই জায়গায় আসে `;`। আমাদের ব্যবহার করা সেমিকোলন দেওয়া কমান্ডগুলো মনে করুন।

`&` মানে পর্দে-কে-পিছে

এই অ্যাম্পারস্যান্ড চিহ্ন `&` আমরা কাজে লাগিয়েছি সম্মুখভূমির সক্রিয় প্রক্রিয়া বা প্রসেসকে পশ্চাৎভূমিতে পর্দার পিছনে নিয়ে যাওয়ার এই চিহ্ন আমরা অনেকবার ব্যবহার করেছি। লিটারালি সেটাই লাগে, কনসোলের গম্ভীর কালো স্ক্রিনের পিছনে চলে গেল, কিন্তু উবে যায়নি, কারন `fg` মারলেই ফেরত আসে।

সুজনদের তেঁতুলপাতা মানে ব্রাকেট, `(` এবং `)`

একলা নয়, একসাথে চলার জন্যে দরকার পড়ে। ঠিক মনে পড়ছে না, কিন্তু এত এত কমান্ড নিয়ে আলোচনা হয়েছে, একটাও ব্রাকেট আসেনি? কে জানে? নয় নম্বর দিনের ২.২ সেকশনে '/etc/sysconfig' আলোচনার সূত্রে আমরা 'finger' কমান্ডটার কথা বলেছিলাম, মনে আছে? কোনো একজন ব্যবহারকারী বিষয়ে তথ্যগুলো তুলে ধরে সামনে? এবার সেই কমান্ডকে আরো সুজনের সঙ্গে মিশিয়ে, ব্রাকেটের তেঁতুলপাতায় বসিয়ে, সময়ের সঙ্গে সঙ্গে পরিবর্তমান আত্মজ্ঞানের উৎস করে তোলা যাক। কমান্ড দিন

```
(finger `whoami`; date)|cat
```

আপনি অবশ্য একে রিডাইরেক্ট করে একটা ফাইলেও রেখে দিতে পারতেন, আপনার আত্মজীবনীর একটা অংশ হিশেবে। আমার আত্মজীবনীর একটা ছেঁড়াপাতা আপনাদের সামনে তুলে ধরা যাক। আমার আলস্যর কথা কেমন হিংস্র রকমে ঘোষণা করে দিয়েছে দেখুন, বিশেষত পরপর 'No Mail. No Plan.' অংশটায় গামা প্যাথোজ যেন থৈ থৈ করছে — নো ওয়ান রাইটস টু দি কর্নেল।

```
Login: dd                               Name: dipankar das
Directory: /home/dd                     Shell: /bin/bash
On since Sat Mar 20 09:08 (IST) on tty1,  idle 0:18
No Mail.
No Plan.
Sun Mar 21 09:26:52 IST 2004
```

ব্রেস বা '{' এবং '}' চিহ্নও একটু আগে আমরা ব্যবহার করেছি, ব্যাশের পাইকিরি ছক বোঝার সময়। আরো অনেক খ্যাঁচসঙ্কুল ব্যবহার আছে এর। ম্যানুয়াল পড়ে জেনে নিন। ব্যাশের পাইকিরি ছকে আরো কী কী অতিচিহ্ন ব্যবহার করে এসেছি দেখুন তো সেটা মনে পড়ছে কিনা? '*', '?', '[' , এবং ']'। এর সঙ্গে আছে রিডাইরেকশন এবং পাইপিং-এর চিহ্ন, মানে '>', '>>', '<' এবং '|'। শব্দযতির কথা বলতে গিয়ে এল লাইন বাড়ানোর চিহ্ন '\'। আছে আদেশ বদল বা কমান্ড সাবস্টিটিউশনের চিহ্ন '`'। বা '$' যা আদেশ বদলও করে, আবার ভ্যারিয়েবলের মানও বোঝায়। আরো আছে, মনে করুন, যে ডিরেক্টরিতে আছেন, সেই মুহূর্তে সেই ডিরেক্টরি বোঝাতে '.'। এই ডিরেক্টরিটা যে ডিরেক্টরির মধ্যে আছে, মানে এর উপরের ডিরেক্টরি বোঝাতে '..'। হোম ডিরেক্টরি বোঝাতে '~'। আর একটা অতিচিহ্ন তো আমরা সেই এক নম্বর দিন থেকেই দেখে আসছি, মানববোধ্য মন্তব্য বা কমেন্টস বোঝাতে, '#', লাইনের শুরুতে যে চিহ্নটা থাকে মানেই সিস্টেম বুঝে যায় এটা মানুষের পড়ার জন্যে। আরো বোধহয় কিছু আছে, ভুলে যাচ্ছি। যাকগে ব্যাশ ম্যানুয়ালের এরকম কোনো ভুল হবেনা। এখন একটু পড়ে দেখে নেওয়া যায়, কিন্তু আমায় দিয়ে সিম্পলি আর হচ্ছেনা। এখন এটা শেষ হলে বাঁচি। দাঁড়ান আর একটা, তাও নিজের মনে পড়েনি, পাতা ওল্টাতে গিয়ে চোখে পড়ল।

শর্তাধীন ক্রিয়ার আদেশ '&&' এবং '||'

ব্যাশকে শর্তাধীন ক্রিয়ার বা কনডিশনাল একজিকিউশনের আদেশ দেওয়ার জন্যে লাগে এই ডাবল অ্যাম্পারস্যান্ড, '&&', এবং ডাবল পাইপ, '||' — মানে বিশেষ একটা কাজ তবেই করবে যদি একটা বিশেষ শর্ত মেটে। এর মধ্যে প্রথমটা আপনাদের চেনা। আগের কাজ শেষ হলে তবে পরেরটায় যাবে, মনে করে দেখুন, আপনার মুন্ডু প্যাকেজ ইনস্টলেশনের সময়ে আমরা কমান্ড দিয়েছিলাম, '`./configure && make && make install`'। যাতে আগের কাজ ঠিক ভাবে সমাপ্ত হওয়ার আগেই পরের কাজে চলে না-যায়। তখন জানতাম না, কিন্তু এখন জানি, একটু টেকনিকাল ভাষায় বলতে গেলে, '&&' চিহ্ন ব্যাশকে জানায় শুধু তখনই পরবর্তী আদেশে যেতে যদি আগের আদেশের নিষ্ক্রমণ স্থিতি বা একজিট স্ট্যাটাস শূন্য হয়। সাফল্যকে শূন্য ভাবে ব্যাশ, মনে আছে তো? ঠিক এর উল্টোটা হল '||'। এই '||' চিহ্নের ডানদিকের আদেশ শুধু তখনই পালন করবে ব্যাশ যদি আগের আদেশটা ব্যর্থ হয়ে থাকে, মানে তার নিষ্ক্রমণ অবস্থা শূন্য না-হয়। ধরুন আপনি জানতে চাইছেন আপনার 'math.text' ফাইলে 'integer' শব্দটা আছে কিনা। আমরা জানি এর কমান্ড 'grep', সেই লাইনগুলোকে নিংড়ে আনে যাতে এই 'integer' শব্দটা আছে এবং স্ক্রিনে দেখিয়ে দেয় লাইনগুলো। আর যদি শব্দটা একবারো না পায়, তাহলে সেই নির্বাক নিঃশব্দ প্রত্যাগমন। শূন্য কমান্ড প্রম্পট খাঁখাঁ করছে স্ফটিক অন্ধকারের দেওয়ালে। তা না করে যদি অন্তত কিছু চান স্ক্রিনে, তাহলে কমান্ড দিন

```
grep 'integer' math.text || echo NO 'integer' found
```

এবার, যদি 'grep' পেয়ে যায় 'integer' শব্দটা তাহলে তো লাইনগুলোই দেখাবে, আর যদি না-পায় তখন দেখাবে তার বার্তা, '`NO 'integer' found`'। এবার মনে হচ্ছে অতিচিহ্ন মোটামুটি নামল। কিন্তু এদের নিয়ে এখনো একটা কুচো সমস্যা আছে। এই অতিচিহ্ন বা মেটাক্যারেকটারগুলো তো বললাম, বিশেষ কিছু অর্থ বহন করে আনে ব্যাশের কাছে, বিশেষ কিছু আদেশ। কিন্তু ধরুন আপনি, কখনো, ওই বিশেষ অর্থে নয়, নিছক একটা চিহ্ন হিশেবেই

ওই বিশেষ মেটাক্যারেকটারটাকে ব্যবহার করতে চান, তখন কী করে করবেন? ধরুন আপনি ব্যাশকে কমান্ড প্রম্পটে দিলেন, ‘`echo An asterix is a *`’। আপনার কী মনে হচ্ছে? খুব সাধারণ একটা লাইন তো, ইকো করেই দেখাবে ব্যাশ। ভুল। করে দেখুন, ‘*’ চিহ্নটার আগের অংশটা, মানে ‘`An asterix is a`’ অংশটা একই থাকবে, কিন্তু তারপরে আসবে, যে ডিরেক্টরিতে দাঁড়িয়ে কমান্ডটা দিয়েছেন, সেই ডিরেক্টরির ভিতরের মোট ফাইলের তালিকা। কারণ, ব্যাশের কাছে ‘*’ মানেই তো তাই। তাহলে এই অতিচিহ্নদের হাত থেকে মুক্তির উপায় কী? কখন একটা ‘*’-কে ‘*’ বলেই বোঝানো যাবে?

১৮।। অতিচিহ্নের থেকে নিস্তার

অতিচিহ্নের অতি-আচার থেকে রেহাই পাওয়ার উপায় একদম বিষে বিষে বিষক্ষয়, মানে আরো কিছু অতিচিহ্ন। দাঁড়ান, সেটার আগে, এই বিন্দু বা ‘.’ চিহ্ন খেয়াল রাখাটা কতটা জরুরি হয়ে পড়ে, একটু আগে বলছিলাম না, তার এইমাত্র দেখা একটা উদাহরণ বলি। আপনাদের জন্যে এটা লেখার আগে আমি একবার করে দেখতে গেলাম। গিয়ে কমান্ড দিলাম ‘`echo An asterix is a *.`’। আরে, আমার বোধভাষ্যি মত, এখানে তো ফাইল তালিকা দেখানোর কথা, ব্যাশ তাই করে বলেই এতদিন জানি। কিন্তু তা না-করে, ব্যাশ হুবহু একদম পুরো কথাটা ইকো করছে। কী হল? তারপর দেখি, ও হরি, আমি ‘*’ না দিয়ে দিয়েছি ‘`*.`’, তাই ব্যাশ সেখানে একে বাড়িয়ে তোলার এক্সপ্যান্ড করার মত কোনো ফাইল তালিকাই পাচ্ছে না, কারণ যেখানে দাঁড়িয়ে দিয়েছি কমান্ডটা সেখানে এমন কোনো ফাইলই নেই যা একটা বিন্দুতে শেষ হচ্ছে। ইন ফ্যাক্ট এরকম কোনো ফাইল আমি আজতক দেখিনি। তাই এক্সপ্যান্ড করতে না-পেরে ব্যাশ চিহ্নটাকে একদম আক্ষরিক আকারেই লিখে দিয়েছে।

এরকম ফাঁকতালে সমাধান বাদ দিন, সত্যিকারের সমাধানের একাধিক উপায় আছে এখানে। এক ব্যাকস্ল্যাশ বা ‘\’। আর দুই, কোট চিহ্ন বা উর্দ্ধকমা। কোট কিন্তু আবার তিনরকম। এক ব্যাককোট, ‘`’, দুই সিঙ্গল কোট ‘'’, আর তিন, ডাবল কোট, ‘"’। এর মধ্যে ব্যাক কোটকে তো আমরা আলাদা করে নিয়েছি আগেই, আদেশ বদল বা কমান্ড সাবস্টিটিউশনের চিহ্ন হিশেবে। এবার এই ব্যাকস্ল্যাশ, এককোট আর জোড়াকোট — ‘\’, ‘'’, ‘"’ — এই তিনটে অতিচিহ্ন ব্যবহার করা হয় অতিচিহ্ন থেকে বাঁচার অতিচিহ্ন হিশেবে। কিন্তু তাদের একজনের থেকে আর এক জনের ব্যবহারের পার্থক্য আছে।

ধরুন, আমাদের ওই তারকাচিহ্ন চেনানোর লাইনটা ভাবুন। ‘`echo An asterix is a *`’। একে যদি হুবহু আমরা স্ক্রিনে দেখতে চাই। এই তিনটে উপায়ের তিনটেই সমান কার্যকরী।

```
echo An asterix is a \*
echo 'An asterix is a *'
echo "An asterix is a *"
```

উপরের তিনটের যে কোনোটাই স্ক্রিনে ফুটিয়ে তুলবে সেই অভিজ্ঞান, ‘`An asterix is a *`’। মানে, ব্যাশ যেখানে ‘*’ চিহ্নটাকে অতিচিহ্ন নয়, স্বাভাবিক একটা সাধারণ চিহ্নের মতই পড়বে। অন্য একটা চিহ্নমালা ‘`*?*`’ দিয়ে এটা করে দেখুন। ‘\’ বেলায় দিতে হবে ‘`\*\?\*`’। ‘'’ হলে ‘`'*?*'`’। আর ‘"’ চিহ্ন হলে ‘`"*?*"`’। এই তিনটে কেসের প্রত্যেকটাতেই আপনি ফল পাবেন এক, ব্যাশ ফুটিয়ে তুলবে ‘`*?*`’। তাহলে এদের তফাতটা কোথায়? তফাত মালুম করার জন্যে আমাদের পুরোনো চেনা তক্তা ‘`$PATH`’ আছে। তাতে একবার করে এই তিনটে পেরেকই মেরে দেখুন। তিনবার এই তিনটে কমান্ড দিন। বোঝা যাবে, এই ‘`$`’ চিহ্নটাকে ভ্যারিয়েবলের মানের সঙ্কেত হিশেবে ভেবে ব্যাশ এটাকে পথনির্দেশের সিস্টেম ভ্যারিয়েবল বলে ভাবছে, তাই তার মানটাকে স্ক্রিনে ফুটিয়ে তুলছে? নাকি, হুবহু গোটাপৃথিবীর স্বপ্নের সবুজের ছবি নিয়ে একটা আক্ষরিক বর্ণসমারোহ করে দেখাচ্ছে? ডলার ছাড়া স্বপ্ন আর কই? ফ্যান্টাসি?

```
echo \$PATH
echo '$PATH'
echo "$PATH"
```

এর প্রথম দুটোয় ব্যাশ ভাববে একদম হুবহু একটা চিহ্ন হিশেবেই। মানে স্ক্রিনে ফুটিয়ে তুলবে, '$PATH'। কিন্তু তৃতীয়টার বেলায় যা পাব তা আমাদের চেনা, আমার মেশিনে সুজে সিস্টেমে 'dd' নামের ব্যবহারকারীর ব্যাশের পথনির্দেশ।

```
/home/dd/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/usr/games:/opt/gn
ome2/bin:/opt/gnome/bin:/opt/kde3/bin:/usr/lib/java/jre/bin:/opt/gnome/bin
```

তার মানে বুঝতে পারছেন, জোড়াকোট বা '"' চিহ্ন '$' চিহ্নকে অতিচিহ্ন বা মেটাক্যারেকটার বলেই ভেবেছে, কিন্তু ব্যাকস্ল্যাশ, '\', বা এককোট, ''' তাকে অতিচিহ্নতা থেকে নিষ্কৃতি দিয়েছে। এই তিনটে চিহ্নের আর একটা আত্মীয়তার জায়গাও আছে। ব্যাকস্ল্যাশ চিহ্ন '\' দিয়ে একটা লাইন থেকে পরের লাইনে কমান্ড টেনে নিয়ে যাওয়ার কথা বলেছি আগেই। এই চিহ্নের পরে এন্টার মারলে নতুন লাইনে যায় প্রম্পট, কিন্তু কমান্ড নেওয়াটা শেষ হয়না। ঠিক সেই একই কাজ করা যায় এককোট ''' আর জোড়াকোট '"' চিহ্ন দিয়েও। একটা এককোট বা জোড়াকোট শুরু করার পর যতক্ষণ না আপনি তার দোসর এককোট বা জোড়াকোট চিহ্নটা মারছেন, ততক্ষণ একটা কমান্ডই চলতে থাকবে। এন্টার মারলে প্রম্পট পরের লাইনে যাবে, কিন্তু কমান্ড শেষ হবেনা। এই তিনটে চিহ্নের তফাতটা একবার জেনে রাখা যাক। তবে আর একবার বলে রাখি, এগুলো ব্যাশ ম্যানুয়াল থেকে একবার মিলিয়েনিন, একদম নিখুঁত বলছি কিনা আমি নিজেই শিওর নই। মানে যেটুকু বলছি সেটা তো সিস্টেমে মিলিয়ে দেখেই বলছি, কিন্তু এর বাইরে আরো কিছু থাকতে পারে, যা আমি নিজেই জানিনা তো লিখব কী।

নিষ্কৃতিচিহ্ন	কাজ করার ধারা
ব্যাকস্ল্যাশ '\'	একটা অতিচিহ্নের এককে কাজ করে, যতগুলো অতিচিহ্নকে আমি আক্ষরিক ভাবে মানে সাধারণ চিহ্নের মত ব্যবহার করতে চাইছি তার প্রত্যেকটার জন্যে একটা করে দিতে হবে। মানে দুটো তারকাচিহ্নের জন্যে, '**', দিতে হবে দুটো, '**'।
এককোট '''	যতটা এলাকাকে দুটো এককোট চিহ্নের মধ্যে রাখা হবে, একটা শুরুর, একটা শেষের, তার মধ্যে গণতন্ত্র হি গণতন্ত্র, সবাই সমান, সবাই জাস্ট চিহ্ন, অতিচিহ্ন বলে তো কিছু হয়না ভাই।
জোড়াকোট '"'	নিষ্কৃতিচিহ্ন হিশেবে রীতিমত বুদ্ধিজীবী, আলাদা আলাদা জায়গায় আলাদা আলাদা বক্তব্য। যতটুকু জায়গাকে দুটো জোড়াকোটের মধ্যে রাখা হবে, শুরুর আর শেষের, সেখানে কিছু চিহ্নকে টেনে নামিয়ে আনবে মাটির পৃথিবীতে, একদম এক ভোটে জীবন, এক গুলিতে মৃত্যু, নরমাল চিহ্ন। কিন্তু তিনটে অতিচিহ্নকে অতিচিহ্নের ওজনেই রেখে দেবে। তারা হল ব্যাকস্ল্যাশ '\', ডলার '$', আর ব্যাককোট '`'।

মা তারা ব্রহ্মময়ী মা, একটু চান করে আসি। আর মাত্র একটা সেকশন। আজ রোববার, একুশে মার্চ বিকেল পাঁচটা ছেচল্লিশ, আর ঘন্টা দুয়েকের মধ্যে মালটা ফিনিশ হবে।

১৯।। নিয়ন্ত্রণ কাঠামো

আমাদের এই বইয়ের শেষ সেকশনে আবার আমরা ফিরে এলাম নিয়ন্ত্রণ কাঠামোর ধারণাটায়। এটাই তো তাই যা কম্পিউটারকে কম্পিউটার করেছে, প্রাককম্পিউটার সমস্ত হিশেবযন্ত্র থেকে আলাদা করে এনেছে। এর ভিতরেই উপস্থিত সচেতন মানুষ। তার সারোগেট, তার প্রতিনিধি — যার অন্য নাম প্রোগ্রাম। আজকের আলোচনার ১০ নম্বর সেকশনে ভন নয়মান কাঠামোর কেন্দ্রীয় নিয়ন্ত্রণ বা 'CC' অংশটাকে ভাবুন। এটাই তো তাই যা স্টোরড প্রোগ্রাম কম্পিউটারের আগে অব্দি অন্য মেশিনগুলোয় ছিলনা। তার আগে অব্দি শুধু র ডেটা বা কাঁচা তথ্যই থাকত মেশিনে। এখন থেকে ইন্সট্রাকশন ডেটা বা আদেশ তথ্যও থাকতে শুরু করল। তিন নম্বর দিনের কম্পিউটারের প্রাগ-ইতিহাসের আলোচনাটা মনে করুন। সেই ডিজিজড এঞ্জেল, ক্লিষ্ট দেবদূত, আডা লাভলেস, তার সেই স্বপ্নগুলো, যাকে আমরা এই শতাব্দীর সবচেয়ে উত্তেজক সায়েন্স ফিকশন বলে ডেকেছিলাম। কেন? মনে করুন সেই কার্ড গুলোর কথা। কার্ডগুলো সংখ্যাকে ধারণ করছে, হিশেবকে ধারণ করছে, কার্ড আর চাকা আর গিয়ার আর ডান্ডা। ডিফারেন্স ইঞ্জিন অব্দি। এল অ্যানালিটিকাল ইঞ্জিন, সেই নতুন ধরনের কার্ডের ধারণা, যা আর হিশেব বা সংখ্যা রাখেনা, রাখে অন্য

কার্ডের কাজের খতিয়ান, তার নিয়ন্ত্রণ। মেশিনের মধ্যেই ঢুকে এল অমেশিন, জড়ের মধ্যে জীবন, সচেতনতা। শূন্য নম্বর দিনের সেই পাঁচ ধরনের উপাদানের কথা মনে করুন। ইনপুট উপাদান, আউটপুট উপাদান, তথ্য প্রসেসিং বা চটকানোর উপাদান, আর তথ্য হোল্ডিং বা ধারণের উপাদান — এই তো গেল চারটে। পাঁচ নম্বরটা? ঠিক মানুষের হাতের তালুতে যা হল, পাঁচ নম্বর আঙুলটা অন্য চারটে আঙুলের সাধারণ সমতল থেকে বেরিয়ে এল, ঘুরে গেল কয়েক ডিগ্রি — ভেবে দেখুন — এই ঘুরে যাওয়া বেঁকে যাওয়া বেরিয়ে যাওয়া বুড়ো আঙুলটাই হল প্রকৃতির উপর অন্য প্রাণীদের থেকে আলাদাতর মানবিক নিয়ন্ত্রণের গোটা ইতিহাসটা। ঠিক তাই ওই পাঁচ নম্বর উপাদানটারও। অন্য চারটে উপাদানকে দেখুন। তারা কী করে? তথ্যকে নিয়ে নাড়াচাড়া করে। যা আসলে কম্পিউটারেরই কাজ। আর পাঁচ নম্বরটা কী করে? সে তথ্য নিয়ে কিছু করেনা। অন্য চারটে উপাদানের কাজকে নিয়ন্ত্রণ করে। এটাই ব্রেক। এটাই ভিশন। অলৌকিকতা। যা সবার থাকেনা, সকলেই কবি নয়, কেউ কেউ কবি। আডার ছিল।

এটাই নিয়ন্ত্রণ কাঠামো। মেশিনের মধ্যেই মেশিনের সেকেন্ড অর্ডার। মেশিনের বাহির মেশিনের অপর মানে মানুষ যখন ঢুকে গেল মেশিনের ভিতর। ঠিক কাজের নিরিখে ভাবা যাক। ধরুন একটা সহজতম ব্যাশস্ক্রিপ্ট। যেখানে নিছক একটা কমান্ডকে সরাসরি ল্যান্ড করিয়ে দেওয়া হয়েছে। 'trashcopy' নামে একটা ফাইল বানান। তাতে নিচের তিনটে লাইন টাইপ করে ঢোকান।

```
#!/bin/bash
cp /etc/trash ~/
echo "Done"
```

প্রথম লাইনটা তো যে ধ্রুবপদ দিয়েছো বাঁধি — ব্যাশপুজোর মন্তর। দ্বিতীয় লাইনটাই হল কাজ, জাস্ট '/etc' ডিরেক্টরির ভিতর একটা '`trash`' নামে ফাইল কপি করে আনা হোম ডিরেক্টরিতে। আর তৃতীয় লাইন হল ব্যাশবাণী, করেছি গো অ্যাক্কেরে করে ফেলেছি। এবার স্ক্রিপ্টটাকে চালনীয় বানান, '`chmod +x trashcopy`'। যেই এক্সিকিউটেবল হল, এবার চালান, '`./trashcopy`'। কী পেলেন?

```
cp: cannot stat `/etc/trash': No such file or directory
Done
```

প্রথম লাইনটা দেখুন, কপি করবে কী বেচারা, ওরকম কোনো নামের ফাইল ওই ডিরেক্টরিতে, কেউ বাপের জন্মে শোনেনি, ফাইলই নেই, তার কপি করবে কী? কিন্তু বুদ্ধির পরাকাষ্ঠাটা পরের লাইনে। সেই কান এঁটো করা হাসি সহ, '`Done`'। হায় গাধা, তুই তো কাজটা পারিসই নি, তাও '`Done`'। এবার ভাবুন, আসলে গাধাটা কে? আমরা তো ব্যাশকে না- গাধা হতে শেখাই নি। আডার সেই বাক্যটা মনে করুন, এমন সব কাজই করতে পারবে মেশিন, যা কী করে করতে হয় আমরা বলে দিতে পারব। আমরা তো শেখাইনি, কী করে ডিসিশন নিতে হয়, কোথায় শুরু করব, কোথায় থামব — এই প্রোপোরশন বোধও শেখাইনি, শেখাইনি কী করে নিজের ভুল সংশোধন করতে হয়। আমরা শেখাই নি, ফাইলটা আছে না নেই সেটাও যাচাই করতে। সেটা তো এমনিতে মেশিন করেনা, আমরা করি। ক্যালকুলেটর বারবার টেপার পরেও যখন কোনো সাড়া পাইনা, আমরা ক্যালকুলেটরের বাইরে, এমনকি বাড়ির বাইরে যাই, গিয়ে ব্যাটারি কিনে আনি, লাগাই, আবার চলে মেশিনটা। সব নিয়ন্ত্রণই আমাদের। অথচ ক্রমে কতদূর নিয়ন্ত্রণ করা যাচ্ছে দেখুন, পরশুদিন তথাগত ফোন করে জানাল ও একটা নতুন ইউপিএস কিনেছে, যেটা সরাসরি তার বিদ্যুৎ কতটা আছে সেই অবস্থাটা কারনেলকে জানিয়ে দিতে পারে। গ্নু-লিনাক্স কারনেল অপশনে দেখবেন এইসব দেওয়া থাকে। এবার ইউপিএস কখন চালু করা কখন বন্ধ করা দরকার সেটাও কারনেল দেখবে। তবে ভারি দাম। অথচ এই নিয়ন্ত্রণটা, আমাদের প্রয়োজনের দূরত্ব অব্দি, ব্যাশকে দিয়ে করানোই যেত। আসুন সেটা কী ভাবে করব তার ফ্লো চার্টটা বানানো যাক। ফ্লো-চার্ট কাকে বলে মনে আছে তো? কী ভাবে কাজটা করব তার ছক এবং গতি এবং ক্রিয়া কাঠামো। আমরা চাই এমন একটা প্রোগ্রাম যে নিজেই দেখে নেবে ফাইলটা আছে কিনা, কপি করা গেল কিনা। নইলে যেটা আমাদের নিজেদের করে নিতে হয়। কেন হলনা? ও দেখি তো ডিরেক্টরিটা, ফাইলটাই আছে কিনা। এবার সেটা ব্যাশ করবে। প্রোগ্রামটার কাঠামোটা হবে এইরকম —

```
if /etc/trash exists,
        copy it to ~/
        print "Done" to the Std. Out.
```

```
Otherwise
        print "file '/etc/trash' does not exist"
        exit
```

ঠিক আমরা মনুষ্যপদবাচ্য বলে পরিচিতরা যেমন যুক্তিকাঠামো বানাই। এবার এই কাঠামোটাকে ব্যাশে আনা যাবে যদি সিদ্ধান্ত নেওয়ার ওই নিয়ন্ত্রণ কাঠামোগুলো ব্যাশের স্ক্রিপ্টিং ল্যাংগুয়েজ দিয়ে নিয়ে আসা যায়। ঠিক এই কাজটাই করে লুপ। ব্যাশে সবচেয়ে প্রাথমিক এবং জরুরি রকমে যে যে লুপের নিয়ন্ত্রণকাঠামো পাওয়া যায় তারা হল, 'if', 'while', 'until', 'for', এবং 'case'। এই প্রতিটি লুপেরই একটা সুনির্দিষ্ট এবং সুস্পষ্ট, অঙ্কের ভাষায় বললে, ফাইনাইট এবং ডেফিনিট, শুরু এবং শেষ আছে, যা দিয়ে মেশিন জেনে যেতে পারে, কোথায় কোন কাজ শুরু হবে, এবং কোথায় কোনটা শেষ হবে।

আসুন প্রথমে এই ফ্লোচার্টটাকে একটা ব্যাশস্ক্রিপ্ট বানাই, যার একটা টেকনিকাল খুঁটিনাটি বুঝতে আপনাকে আরো দুতিনটে প্যারা ওয়েট করতে হবে। ওঃ ঈশ্বর, ওয়েটটা এখন আর এমনকি সেকশনের নয়, দিনের তো নয়ই, জাস্ট কয়েকটা প্যারার। এই স্ক্রিপ্টটা বানান 'trashcopy1' নামে একটা ফাইলে। তারপর তাকেও ঠিক আগের মতই করে চালনীয় করতে হবে এবং চালাতে হবে।

```
#!/bin/bash
if test -f /etc/trash
then
        #file exists, so copy and display
        cp /etc/trash ~/
        echo "Done"
else
        #file does not exist, so display error
        echo "There is no file '/etc/trash'"
        exit
fi
```

এই কাঠামোটা খেয়াল করুন। লুপটার গঠনটা হল 'if … then … else … fi'। 'if' দিয়ে শুরু, 'fi' দিয়ে শেষ। মধ্যে মধ্যে '#' চিহ্নটা দিয়ে যথারীতি মানবপাঠ্য এবং মানববোধ্য মন্তব্য আনা হয়েছে, ওই লাইনগুলো অব্যাশনীয়। মন্তব্যগুলো পড়ে দেখুন, সহজেই বুঝতে পারবেন, কী বলা এবং করা হয়েছে। শুধু একটা লাইন বুঝতে, বললামই তো, কয়েক প্যারা অপেক্ষা করতে হবে। এই স্ক্রিপ্টটা একদম কপিবুক স্ক্রিপ্ট, গাভাস্কারের ব্যাটিং-এর মত। সব নিয়ম মানা হয়েছে, সবসময় এত মানা হয়না। আর আগেই তো বলেছি, স্ক্রিপ্টের নামের এক্সটেনশন থাকা উচিত, '.sh'। আইন মেনে এই স্ক্রিপ্টটার নাম হওয়া উচিত ছিল, 'trashcopy1.sh'। আমার আর করা হয়না এসব। আসলে এগুলো তো কাজের স্ক্রিপ্ট না, জাস্ট আপনাদের স্ক্রিপ্ট পড়ার অভ্যেশ আনার জন্যে বানানো। আর আমার সব স্ক্রিপ্ট সবই থাকে হোম ডিরেক্টরির ভিতর '~/bin' বলে একটা ডিরেক্টরিতে। তাই বোঝা না-বোঝাটা এসব সমস্যা হয়না। কিন্তু ভালো না এসব ভালো না, বলেছেন শাক্যমুনি, বড়দাদের বাতেলা শুনতে শুনতেই বড় হওয়ার অভ্যেশ করাটাই হল ডিফল্ট। এবার চালান স্ক্রিপ্টটাকে। কী পেলেন?

```
There is no file '/etc/trash'
```

দেখছ কাকা, আমাদের স্ক্রিপ্টও বলতে শুরু করেছে, মহারাজ, আজ মনে হচ্ছে আমাদের বয়স হয়েছে। জিতা রহো বাচ্চা। স্ক্রিপ্টটার শরীরে ইনডেন্টগুলো দেখুন, এক একটা কাজের ব্লককে ইনডেন্ট করে ডানে সরিয়ে এক এক সঙ্গে আনা হয়েছে, এটাও স্ক্রিপ্টের কাজ করার নয়, স্ক্রিপ্ট পড়ে বোঝার সুবিধের স্বার্থে। একটা ব্লক 'then' অংশটার, আর একটা ব্লক 'else' অংশটার। কেন, বুঝতে পারছেন?

এবার ওই হিব্রু ভাষায় লেখা লাইনটা একটু মাপা যাক। যে মূল জিনিষটা ব্যবহার করা হয়েছে সেটা 'test', একটা প্রোগ্রাম। পাবেন ম্যানপেজের এক নম্বর সেকশনে। দেখুন ম্যানপেজের গৌরচন্দ্রিকাই বলে দিচ্ছে, টেস্ট কী করে — ফাইলের প্রকার পরখ করে এবং আলাদা আলাদা মানের ভিতর তুলনা করে, 'test - check file types and compare values'। এই ফাইল মানে যে কোনো রকম ফাইল, কোন ফাইল কী ধরনের সেটা যাচাই করে 'test'। এবং তুলনা করে একটা মানের সঙ্গে অন্য মানের, তারা সমান না বড় না ছোট না শূন্য এইসব দেখে। মান বলতে আঙ্কিক

ভ্যারিয়েবল বা স্ট্রিং ভ্যারিয়েবল মানে চিহ্ন-সমাহার দিয়ে তৈরি ভ্যারিয়েবলের মান। ম্যানপেজ থেকে 'test' প্রোগ্রামের অপশনগুলো পড়ে দেখুন, '-f' অপশনটা যাচাই করে দেখে কোনো রেগুলার ফাইল আছে কিনা। এবার আমাদের দেওয়া নাম আর ধাম দেখে ব্যাশ 'test' কাজে লাগিয়ে যাচাই করে নিচ্ছে ওই '/etc' ডিরেক্টরিতে 'trash' নামে কোনো ফাইল আছে কিনা। এই 'trashcopy1' ফাইলে আমরা কোনো ফাইল ওই নামে ওই ধামে আছে কিনা যাচাই করার জন্যে আদেশ দিয়েছিলাম, 'test -f /etc/trash'। এই আদেশটা এর থেকে একটু অন্যরকম আকারেও দেওয়া যেত, '[-f /etc/trash]'। তখন স্ক্রিপ্টটার আকার দাঁড়াত

```
#!/bin/bash
if [ -f /etc/trash ]; then
        #file exists, so copy and display
        cp /etc/trash ~/
        echo "Done"
else
        #file does not exist, so display error
        echo "There is no file '/etc/trash'"
        exit
fi
```

চালিয়ে দেখুন, একই ভাবে চলবে। 'test' আদেশের এই আকারটা অনেক বহুলব্যবহৃত। যেমন বললাম, এই টেস্ট প্রোগ্রামটা ব্যবহার করা যায় দুটো ভ্যারিয়েবলের ভিতর তুলনায়, বা একটা ভ্যারিয়েবল এবং একটা কন্সটান্ট বা স্থির রাশির ভিতরে তুলনায়। যেমন ধরুন আমরা যদি আদেশ দিই '["$number -eq 5]', টেস্ট তখন যাচাই করে দেখবে 'number' নামক ভ্যারিয়েবলের মান '5'-এর সমান কিনা। সত্যিই আমার কাজে লাগে এমন একটা স্ক্রিপ্ট তুলে দিই এখানে, যা এই 'test' ব্যাপারটা ব্যবহার করেছে। এখানে যেটা তুলে দিচ্ছি সেটা অবশ্য আমার আদত স্ক্রিপ্টটা নয়। তার সঙ্গে এমন কিছু যোগ করেছি যাতে বুঝতে সুবিধে হয় কী করে স্ক্রিপ্টটা কাজ করছে। কিছু মন্তব্য যোগ করেছি ওই '#' চিহ্ন দিয়ে, সেটাও একই কারণে। আমার মেশিনে তিনটে গ্নু-লিনাক্স অপারেটিং সিস্টেমের জন্যে স্ক্রিপ্টটা লেখা, সুজে, স্ল্যাকওয়ার আর ম্যানড্রেক। কী নাম তাদের বা ঠিক তিনটেই কিনা তাতে অবশ্য কিছু এসে যায়না, দেখতেই পাবেন। এই প্রত্যেকটাতেই একজন করে ইউজার 'dd' আছে, মানে আমি। নানা সময়ে নানা কিছু করে দেখার জন্যে আমি আমার ব্যাশের লোকাল কনফিগারেশন ফাইলগুলোয় নানা বদল করি, কিন্তু সেই বদলগুলো মাঝে মাঝেই চূড়ান্ত ফ্লপ করে, তখন ম্যাও সামলানোর দরকার পড়ে। তাই এই কনফিগারেশন ফাইলগুলো, ব্যাকআপ করার দরকার পড়ে প্রতিটি সিস্টেমের '/home/dd' ডিরেক্টরি থেকে। লোকাল কনফিগারেশন ফাইল মানে '~/.bashrc' আর '~/.profile'। আমরা এখানে গোটা স্ক্রিপ্টটার পাশে বাঁদিকে পরপর বাংলায় লাইন নম্বরগুলো দিয়ে গেলাম, যাতে আলোচনা করার সুবিধে হয়।

০১	#!/bin/bash
০২	rm /mnt/arkive/bkp/bash/bkp.*.*
০৩	#remove old backups
০৪	counter=1
০৫	#initialize the counter
০৬	for directory in / /mnt/slackware /mnt/mandrake
০৭	#first 'for' loop starts here
০৮	#for three systems in the machine, suse, slackware and mandrake,
০৯	# mounted on /, /mnt/slackware and /mnt/mandrake
১০	do cd $directory
১১	#going into the three root directories, one by one
১২	echo "Now probing $directory"
১৩	# a check if the script is working properly
১৪	if [-d $directory/home/dd];then
১৫	#first 'if' starts here

```
১৬          echo "$directory/home/dd found"
১৭                  if [ -f $directory/home/dd/.bashrc ];then
১৮  #second 'if' starts here
১৯                  echo "'.bashrc' found in $directory/home/dd"
২০                  else echo "No '.bashrc' found in $directory/home/dd"
২১                  fi
২২  #second 'if ends here'
২৩                  if [ -f $directory/home/dd/.profile ];then
২৪  #third 'if' starts here
২৫                  echo "'.profile' found in $directory/home/dd"
২৬                  else echo "No '.profile' found in $directory/home/dd"
২৭                  fi
২৮  #third 'if' ends here
২৯          cd $directory/home/dd
৩০  #going into the '/home/dd' directory of the three systems, one by one
৩১                  for file in `ls .bashrc .profile`
৩২  #second 'for' starts here
৩৩                  do cp $file /mnt/arkive/bkp/bash/bkp$file.$counter
৩৪                  done
৩৫  #second 'for' ends here
৩৬          else echo "No $directory/home/dd found"
৩৭          fi
৩৮  #first 'if' ends here
৩৯  counter=$(expr $counter + 1)
৪০  #counter is incremented by 1 after every turn of first 'for' loop
৪১  echo "Counter is $counter"
৪২  #another kind of check
৪৩  done
৪৪  #first 'for' ends here
৪৫  cd /home/dd
৪৬  #back home real home
```

এবার দেখা যাক লাইন বাই লাইন। শুধু মূল স্ক্রিপ্টে যে লাইনগুলোয় মন্তব্য আছে সেগুলোকে বাদ দিয়েছি এই আলোচনা থেকে। যারা মূল স্ক্রিপ্টের মন্তব্যটুকু পড়েই কাজ করার রকমটা বুঝে যেতে পারছেন, তাদের আর এই লাইন ধরে ধরে আলোচনাটা পড়ার দরকারই নেই।

০১। এই লাইনটা সেই ধ্রুবপদ, ব্যাশস্ক্রিপ্টের বাধ্যতা।

০২। এই লাইনে ব্যাশকে বলা হচ্ছে, বাবা ব্যাশ, তুমি '/mnt/arkive/bkp/bash' ডিরেক্টরি থেকে কনফিগারেশন ফাইলের পুরোনো ব্যাকআপ ফাইলগুলো উড়িয়ে দাও।

০৪। গোটা স্ক্রিপ্ট জুড়ে আমরা যে কাউন্টারটা ব্যবহার করব, তাকে শূন্য করে নিচ্ছি, পুরো প্রক্রিয়ার ধাপগুলোকে যাতে গুনতে গুনতে যাওয়া যায়, ওই গোনা থেকে তৈরি সূচক আমরা পরে আমাদের ব্যাকআপ করা ফাইলগুলোর নামে ব্যবহার করব, সিস্টেম থেকে সিস্টেমে ফাইলগুলোকে আলাদা করার জন্যে। এই মুহূর্তে এই 'counter' ভ্যারিয়েবলের মান মানে '$counter' হল '1'। পরে ধাপে ধাপে বাড়বে, স্ক্রিপ্টের ৩৯ নম্বর লাইনে গিয়ে। প্রতিবার বাড়বে এক করে।

০৬। এই লাইনটা থেকে শুরু হচ্ছে প্রথম 'for' লুপের নিয়ন্ত্রণ কাঠামো। দেখুন এখানে তিনটে ডিরেক্টরির নাম দেওয়া আছে, '/', '/mnt/slackware', এবং '/mnt/mandrake'। মানে যেখানে যেখানে আমার মেশিনের তিনটে গ্নু-লিনাক্স সিস্টেমের তিনটে রুট পার্টিশন মাউন্ট করা আছে। এই ব্যাশ স্ক্রিপ্টটা বানানো সুজে সিস্টেমের ভিতরে দাঁড়িয়ে, যেখানে, '/mnt' ডিরেক্টরিতে দুটো সাবডিরেক্টরি '/mnt/slackware' আর '/mnt/mandrake' জুড়ে স্ল্যাকওয়ার আর ম্যানড্রেকের রুট পার্টিশন দুটো মাউন্ট করা আছে। এখানে দেওয়া ছকটা আমাদের গোটা বই জুড়ে ব্যবহৃত পুরোনো ছক থেকে একটু আলাদা, সেখানে কোনো ম্যানড্রেকই ছিলনা। এর মধ্যে আমার হার্ডডিস্কটা বদলে গেছে, মনে আছে? আর আগে তিনটে সিস্টেমের কথা আনতে চাইনি, জটিলতাটা বড্ড বেশি বেড়ে যাচ্ছিল বলে। এখন আমরা অনেকটা অভ্যস্ত হয়ে গেছি। এবং এই স্ক্রিপ্টটাকে আমার মেশিনের ব্যবস্থা থেকে একদম আলাদা ভাবেই পড়ুন। এমনকি সেই মেশিনে তিনটে ব্যবস্থা যদি নাও থাকে তাহলেও কিছু এসে যাবেনা, পরের গুলো সিস্টেম আর পাবেনা, কিন্তু গ্নু-লিনাক্স সিস্টেম থাকলে '/' থাকবেই, তার মানে অন্তত একবার স্ক্রিপ্টটা তার লুপে ঘুরে আসবেই। এবার এই 'for' লুপটার কাঠামো খেয়াল করুন। আমরা 'directory' বলে একটা ভ্যারিয়েবল তৈরি করছি, যার তিনটে আলাদা আলাদা মান হতে পারে '/', '/mnt/slackware', এবং '/mnt/mandrake'। এবার এই 'for' লুপটা ব্যাশকে বলছে তুমি পরপর এক একটা করে '$directory' মানে 'directory' ভ্যারিয়েবলের মান তোলো এবং সেই মান অনুযায়ী স্ক্রিপ্টের পরের লাইনগুলোয় দেওয়া আদেশগুলো চালাও। যেই একটা মানের জন্যে গোটা স্ক্রিপ্টের সবকটা আদেশ পালন করা শেষ হবে, তখন পরের মানটা নাও এবং আবার গোড়া থেকে শুরু করো। মনে পড়ছে আডার সেই কার্ডের কথা? কোনো মিল পাচ্ছেন?

১০। এই বার কাজ শুরু হচ্ছে। প্রথমে 'cd' করে সেই ডিরেক্টরিতে চলে যাও যার মান তোমার কাছে এই মুহূর্তে ভরা আছে 'directory' ভ্যারিয়েবলের জন্যে বরাদ্দ জমিতে। লুপটা যদি প্রথমবার চলছে তো এই মুহূর্তে '$directory' হল '/', তার মানে দাঁড়াচ্ছে — '`cd /`' আদেশটা পালন করো। তার মানে ব্যাশের ওয়ার্কিং ডিরেক্টরিটা বদলে গেল, খেয়াল করুন, এই মুহূর্তে সেটা '/'। লুপের দ্বিতীয় ঘোরায় এই জায়গায় এসে ব্যাশ যাবে '/mnt/slackware' ডিরেক্টরিতে, তৃতীয় ঘোরায় এসে যাবে '/mnt/mandrake' ডিরেক্টরিতে।

১২। এই লাইনটা মূল স্ক্রিপ্টে ছিলনা, আপনাদের জন্যে বানানো ভার্শনটায় ঢোকানো, অনেকটা মাছ ধরার ছিপের ফাতনার মত, তলায় তলায় ঘটনা কোন দিকে এগোচ্ছে তার উপরে নজরদারি করার একটা কৌশল। ভেবে দেখুন, লুপের প্রতি ঘোরায় একবার করে ব্যাশ এই লাইনে আসবে এবং ফুটিয়ে তুলবে সে কোন ডিরেক্টরিতে আছে, তিনবার তিনটে আলাদা নাম আসবে। স্ক্রিপ্টটা যে ঠিকঠাক এগোচ্ছে সেটা বোঝা যাবে।

১৪। এই প্রকৌশলটা তো আমরা শিখলাম এইমাত্র এই সেকশনেই। এটা বিচার করে দেখছে '$directory/home/dd' নামের কোনো ডিরেক্টরি আছে কিনা আদৌ। এই মুহূর্তে '$directory' হল '/', তার মানে আদতে আমরা '/home/dd' নামের ডিরেক্টরি খুঁজছি। লুপের পরের ঘোরাগুলোয় এই নামটা বদলে যাবে। এর ঠিক পরের বার নামটা হবে '/mnt/slackware/home/dd'। তার পরের বার কী হবে বলুন তো? এই লাইনেই শুরু হচ্ছে আর একটা নিয়ন্ত্রণ কাঠামো, 'if'। আমরা এই স্ক্রিপ্টে এই 'if' কাঠামোটা তিনবার ব্যবহার করেছি, তার প্রথমবার এই শুরু হল।

১৬। লাইন নম্বর ১৫-তে যদি ব্যাশ পরখ করে পায় যে আছে, আছে, অমন একটা ডিরেক্টরি আছে বৈকি, সঙ্গে সঙ্গে সেটা আমাদের জানিয়ে দেওয়ার ব্যবস্থা। নির্বাক ব্যাশকে একটু কথা বলতে শেখানো, যাতে তার পেটে পেটে কী ঘটছে সেটা আমরা জানতে পারি। এই লাইনটাও মূল স্ক্রিপ্টে ছিলনা, আপনারা চালাতে গেলে যাতে মূল কাঠামোটা ধরতে পারেন, তার জন্যে যোগ করা।

১৭। এই এলো দ্বিতীয় 'if'। খেয়াল করুন, প্রথম 'if' এখনো শেষ হয়নি, তার পেটের মধ্যে এই দ্বিতীয় 'if'। একে টেকনিকাল ভাষায় বলে নেস্টেড লুপ। লুপের বাচ্চা লুপ। তার মানে প্রথম 'if' যদি মেলে তবেই আমরা এই দ্বিতীয় 'if' নিয়ন্ত্রণে ঢুকছি। প্রথম 'if'-এর শর্তটাই যদি না মেলে তাহলে এই দ্বিতীয় 'if'-এ আমরা আদৌ ঢুকছি না। মানে '/home/dd' ডিরেক্টরি না-থাকলে দ্বিতীয় 'if' আমরা বিচার করছিই না, কারণ করার দরকারই পড়ছে না। দ্বিতীয় 'if' পরখ করে দেখছে '/home/dd' ডিরেক্টরিতে কোনো '.bashrc' ফাইল আছে কিনা। ভাবুন,

ডিরেক্টরিটাই না-থাকলে এটা পরখ করার দরকারই পড়ত না, সবাই তো আর চেশায়ার বিড়াল নয় যে আপনি না-থাকলেও আপনার হাসিটা রয়ে যাবে।

১৯। এই লাইনে জাস্ট ফাইলটা পাওয়া গেলে সেটা জানাতে বলা।

২০। যদি না পাওয়া যায়, তাও বলে দাও বাবা ব্যাশ।

২১। ইনি হলেন 'if'-এর শীর্ষাসন, মানে 'if' শেষ হওয়ার ইঙ্গিত। কোন 'if' শেষ হল এখানে? ঠিক এর আগেই যে 'if' শুরু হয়েছে, মানে দ্বিতীয় 'if'। 'fi' আমাদের জানাল, ১৭ নম্বর লাইনে বিসমিল্লা হওয়া এই স্ক্রিপ্টের দ্বিতীয় 'if' নিয়ন্ত্রণ কাঠামোটার ইন্তেকাল হল এই ২১ নম্বর লাইনে এসে।

২৩। এখান থেকে ২৭ নম্বর লাইন অব্দি হল এই স্ক্রিপ্টের তৃতীয় 'if' নিয়ন্ত্রণ। আর সব কিছু হুবহু এক দ্বিতীয় 'if'-এর সঙ্গে, শুধু এবার ব্যাশ খুঁজছে '.profile', এবং তার হ্যাঁ-সংবাদ বা না-সংবাদ জানিয়ে দিচ্ছে আমাদের। ২৭ নম্বর লাইনে এসে শেষ হচ্ছে এই তৃতীয় 'if'। প্রথম 'if' কিন্তু এখনো চলছে। মানে, তৃতীয় 'if'-ও একটা নেস্টেড ইফ। এবং এই প্রথম 'if' আবার চলছে প্রথম 'for' লুপের পেটের ভিতরে বসে। মনে আছে? প্রথম 'for' থেকেও আমরা এখনো বেরোইনি, যাতে ঢুকেছিলাম আমরা ০৬ নম্বর লাইনে।

২৯। এই লাইনে এসে ব্যাশ আবার তার ওয়ার্কিং ডিরেক্টরি বদলাচ্ছে। লাইন নম্বর ১০-এ ব্যাশ ঢুকেছিল '/' ডিরেক্টরিতে। তারপর '/home/dd' ডিরেক্টরির ফাইলের খবরাখবর নিচ্ছিল সেখানে বসেই। এবার ব্যাশ ঢুকছে '/home/dd' ডিরেক্টরিতে। এখন তার ওয়ার্কিং ডিরেক্টরি '/home/dd'। প্রথম 'for' লুপের দ্বিতীয় বার ঘোরায় এই লাইনে এসে ব্যাশ ঢুকবে '/mnt/slackware/home/dd' ডিরেক্টরিতে। শেষবার এই লাইনে এসে ব্যাশ কোন ডিরেক্টরিতে ঢুকবে বলুন তো?

৩১। এই আদেশ বদল বা কমান্ড সাবস্টিটিউশন আমাদের চেনা, '/home/dd' ডিরেক্টরিতে '.bashrc' বা '.profile' ফাইল যা খুঁজে পাবে ব্যাশ 'ls' করে, তাদের নিয়ে শুরু হচ্ছে এই দ্বিতীয় 'for' লুপ। এটাও নেস্টেড লুপ, কারণ প্রথম 'for' এখনো চলছে। চলছে প্রথম 'if' নিয়ন্ত্রণও। প্রথম 'for' লুপের ঘোরার সম্ভাব্য সর্বোচ্চ সংখ্যা তিন, এই দ্বিতীয় 'for' ঘোরার সম্ভাব্য সর্বোচ্চ সংখ্যা দুই, কারণ ফাইল থাকতে পারে দুটো। আর যদি একটা ফাইলও না-থাকে, তাহলে তো ল্যাটা চুকেই গেল। এই লুপ পয়দা হওয়া মাত্রই মরে গেল, ব্যাশ এগিয়ে যাবে স্ক্রিপ্টের পরবর্তী লাইনে। দেখুন, এই দ্বিতীয় 'for' লুপে ভ্যারিয়েবলের নাম হল 'file'। এই 'file' ভ্যারিয়েবলের তাই মান হতে পারে মাত্র দুটো, হয় '.bashrc' নয় '.profile'। ধরুন দুটো ফাইলই পেয়েছে ব্যাশ। তাহলে প্রথমবার ঘোরার সময় '$file' বা 'file' ভ্যারিয়েবলের মান কত? অবশ্যই '.bashrc'। দ্বিতীয়বার ঘোরার সময় '$file' দাঁড়াবে '.profile', যদি দুটো ফাইলই পাওয়া যায়।

৩৩। মজার কথা দেখুন, আসলে এই গোটা স্ক্রিপ্ট জুড়ে ব্যাশ যে কাজটা করবে সেটা আসলে এই লাইনটুকুই। যদি '.bashrc' আর '.profile' ফাইল দুটো পায়, বা একটাও পায় তাদের ভিতর থেকে, সেই ফাইল এবার ব্যাশ কপি করে দেবে '/mnt/arkive/bkp/bash' ডিরেক্টরিতে, যা অবশ্যই আগে থেকে বানানো আছে সিস্টেমে। আর ফাইল যদি না-পেয়ে থাকে সেটা তো ব্যাশ আগেই জানিয়ে দিয়েছে আমাদের। এবং কপি করার সময় ফাইলটার নাম বদলে দেবে। বদলের ছকটা হল 'bkp$file.$counter'। মানে নামটার আগে একটা 'bkp', পরে '.$counter'। এবার খেয়াল করুন, '$counter' এখন কত আছে। এটা প্রথম 'for' লুপের প্রথমবার ঘোরা চলছে, তার মানে '$counter' এখন '1'। আর '$file' এই মুহূর্তে '.bashrc'। তাহলে ব্যাক-আপ করা ফাইলের নাম দাঁড়াবে, তিনটে অংশ মিলিয়ে 'bkp.bashrc.1'। এই কপি করে চলা জুড়েই চলছে দ্বিতীয় 'for' লুপ।

৩৪। দ্বিতীয় 'for' লুপ শেষ হল, শেষ হওয়ার সিগনালটা আমরা চিনি, 'done'। প্রথম 'for' লুপ কিন্তু এখনো চলছে। এবং প্রথম 'if' নিয়ন্ত্রণও।

৩৬। এই আমরা এলাম আমাদের প্রথম 'if' নিয়ন্ত্রণের শেষ অংশে। ১৪ নম্বর লাইনে শুরু হওয়া 'if' নিয়ন্ত্রণের ভিতর আমরা এতটা সময় ধরে এগিয়েছি এই ধারায় যে '/home/dd' ডিরেক্টরিটা পাওয়া গেছে। কিন্তু যদি ডিরেক্টরিটা না পাওয়া যায় আদৌ? তাহলে কী হবে? এবার সেই দ্বিতীয় ধারা, ব্যাশ আমাদের পস্ট জানিয়ে দেবে, ওসব পাওয়া ফাওয়া যায়নি মামা, কপি কী করব, ডিরেক্টরিই নেই, এসব আওফাও কাজ দাও কেন?

৩৭। এসব বলার পরে ব্যাশের আর কী করার থাকে 'fi' ছাড়া? মানে টাটা, মানে প্রথম 'if' নিয়ন্ত্রণ শেষ হল এই লাইনে এসে।

৩৯। এই কলকব্জাটা তো আমরা ধরে ধরে আলোচনা করেছি আগেই, এক করে বাড়িয়ে দেওয়া 'counter' নামে ভ্যারিয়েবলটাকে। তার মানে খেয়াল করুন, এবার '$counter' হয়ে গেল '2'। প্রথম 'for' লুপ পরের বার ঘোরায় আসব যখন আমরা তখন ব্যাক-আপ করা ফাইলে যোগ হবে '2'। ৩৩ নম্বর লাইনের সঙ্গে মেলান। তার মানে সুজের বেলায় হবে '1'। স্ল্যাকের বেলায় '2'। আর ম্যানড্রেকের বেলায় '3'। যদি না পাওয়া তাহলে তো সবচেয়ে ভালো, ব্যাক-আপ করতেই হবেনা।

৪১। এটাও একটা সেই দুধেভাতে-গার্ড। বোঝার সুবিধের জন্যে লাগানো।

৪৩। এতটা পথ পেরোলে লুপ খতম বলা যায়? প্রশ্নটা খুব সহজ এবং উত্তর-ও তো জানা। নিজেই করুন, নিজেই উত্তর দিন। কিন্তু একটা ঘোরা খতম মানেই পরের ঘোরার শুরু। প্রথম 'for' লুপ শুরু হয়েছিল ৬ নম্বর লাইনে। সেখানে 'directory' ভ্যারিয়েবলের যে তিনটে মান দেওয়া হয়েছিল, তার পরের মানটা নিয়ে শুরু হবে এবারের ঘোরা। এরকম চলবে, যতক্ষণ না সবকটা মান শেষ হয়।

৪৫। আপনি কোথায় বসে পড়ছেন? যদি বাড়িতে হয়, তাহলে বাইরে বেরিয়ে গিয়ে এই প্রথম গরমের রাস্তায় সিভিল ইঞ্জিনিয়ার আর আওয়ারা কুত্তাদের সঙ্গে মোলাকাত করে আসুন, নিজেই বুঝতে পারবেন এই লাইনটার মানে। (কার্টসি টেনিদা)।

তার মানে, 'for … do … done' লুপও শেষ হল, এই স্ক্রিপ্টটা বুঝতে গিয়ে। এবার দেখা যাক আর একটা ধরনের নিয়ন্ত্রণকাঠামো, 'while ... do ... done' লুপ। শুরু করা যাক একটা স্ক্রিপ্ট দিয়ে। এখন আমরা স্ক্রিপ্ট পড়ে অনেকটাই অভ্যস্ত হয়ে এসেছি। এই স্ক্রিপ্টটার আমি নাম দিয়েছিলাম 'whileshow'। যে নামই দিন, সেটাকে চালনীয় বানান এবং চালান।

```
#!/bin/bash
while true; do
        echo "Press Ctrl-C to quit."
done
```

চালানো মাত্র কী দেখলেন, কোটি কোটি লাইন নেমে যাচ্ছে স্ক্রিনে, '`Press Ctrl-C to quit.`'। এবং সত্যিই এই প্রবাহ থেকে মুক্তি পাওয়ার একমাত্র উপায় ওই '<Ctrl><C>' টাইপ করা। এখানে ব্যাশকে আমরা ব্যবহার করিয়েছি 'test'-এর মতই আর একটা প্রোগ্রাম, তার নাম 'true'। এই প্রোগ্রামটা খুব মজার, এই নিয়ে গ্নু-লিনাক্স জগতে অনেক চ্যাংড়ামিও আছে। ট্রু সবসময়েই ট্রু, যদিনা আপনি যে প্রক্রিয়াটাকে দিয়ে 'true' প্রোগ্রামটা ডেকে এনেছেন, সেই প্রক্রিয়াটাই বন্ধ করে দেন। এখানে আমরা যা করছি '<Ctrl><C>' টাইপ করে। এই 'true'-এরও ম্যানুয়াল পাবেন ম্যান পেজের সেকশন একে। পড়ে দেখুন। ট্রু কিছুই করেনা, কিন্তু এই না-করাটা অত্যন্ত সাফল্যের সঙ্গে করে — না না, এটা আমার নয়, ম্যানপেজের অভিমত। এই 'while ... do ... done' লুপটা দিয়ে আর একটা স্ক্রিপ্ট বানানো যাক, আমি নাম দিয়েছিলাম 'whileshow1'।

```
#!/bin/bash
c=0
#initializing counter
while [ "$c" -le 10 ]; do
        echo "Now counter is: $c"
        c=$(expr $c + 1)
        # incrementing counter by one
        sleep 1
done
```

অনেকগুলো স্ক্রিপ্ট তো হল, এবার নিজে বোঝার চেষ্টা করুন তো, কী করছে এখানে? ধাপে ধাপে বাড়াচ্ছি আমরা কাউন্টার মানে 'c' ভ্যারিয়েবলটা, আর সেটাকে ফুটিয়ে তুলছি স্ক্রিনে, আর তারপর এক সেকেন্ড করে ব্যাশকে জিরোনোর ফুরশত দিচ্ছি প্রত্যেক ধাপেই। এর মধ্যে 'test' দিয়ে যে পরখ করে নেওয়া সেটাকে খেয়াল করুন, শর্তটা হল, '"$c" -le 10'। মানে কাউন্টার ভ্যারিয়েবলটার মান '-le 10' মানে '10'-এর ছোট বা সমান কিনা, সেটাই দেখে

নিচ্ছে ব্যাশ। আগেই তো বলেছি 'test' কমান্ডের ম্যানুয়াল পড়ুন, সবগুলো খুঁটিনাটিই পেয়ে যাবেন। কাউন্টার ভ্যারিয়েবলটাকে আমরা জোড়াকোটের ভিতরে দিয়েছি, এটা বাধ্যতামূলক নয়, কিন্তু সুবিধাজনক, অনেক গোলযোগ কম হয়। এবার এই স্ক্রিপ্টটাকে চালান। পাবেন নিচের এই লাইনগুলো। কিন্তু প্রতিটা লাইন আসবে পরের লাইনের এক সেকেন্ড পরে।

```
Now counter is: 0
Now counter is: 1
Now counter is: 2
Now counter is: 3
Now counter is: 4
Now counter is: 5
Now counter is: 6
Now counter is: 7
Now counter is: 8
Now counter is: 9
Now counter is: 10
```

এক সেকেন্ড পরে পরে লাইনগুলোর অবতীর্ণ হওয়াটা আমরা ঘটাচ্ছি ব্যাশকে ঘুম পাড়িয়ে পাড়িয়ে। কমান্ডটা হল 'sleep'। ম্যান পড়ার নেশা করো ম্যান, খেজুরগাছে হাড়ি বাঁধো ম্যান। তবে একটা ছুটকো চানস আছে গড়বড়ের। সব সিস্টেমে 'sleep' নাও পেতে পারেন, তখন খুঁজুন 'usleep'। স্ক্রিপ্টটা সেটা দিয়েও বানিয়ে নিতে পারবেন। এইমাত্র করা প্রোগ্রামটাই অন্য একটা লুপ দিয়েও করা যেত, 'until ... do ... done'।

```
#!/bin/bash
c=0
until [ "$c" -ge 10 ];do
        echo "Now counter is: $c"
        c=$(expr $c + 1)
        sleep 1
done
```

এখানে খেয়াল করুন, আঙ্কিক যুক্তিটা একটু উল্টে দেওয়া হয়েছে, 'test' এখন পরখ করছে, '"$c" -ge 10' কিনা, মানে 'c' ভ্যারিয়েবলের মান দশের চেয়ে বড় কিনা? এটা কী হল? ভাবুন, লুপটাকে, হোয়াইল হল 'যখন', আর আনটিল হল 'যতক্ষণ না'। যখন আমি দশের ছোট, তার মানেই তা যতক্ষণ না আমি দশের বড় বা সমান।

এখন আমাদের বাকি আর একটাই লুপের কথা, সেটা হল 'case … in … done'। এর একটা উদাহরণ দিয়ে এসেছি আগেই, 'xpick' নামে, নয় নম্বর দিনে।

```
#!/bin/bash
echo "Choose a number to pick your Window Manager"
echo ""
echo "1.  KDE"
echo ""
echo "2. GNOME"
echo ""
echo "3. FLUXBOX"
echo ""
echo "4. BLACKBOX"
echo ""
read NUMBER
case $NUMBER in
   1) export WINDOWMANAGER=startkde ;;
   2) export WINDOWMANAGER=gnome-session ;;
   3) export WINDOWMANAGER=fluxbox ;;
   4) export WINDOWMANAGER=blackbox ;;
   *) echo ""; echo "Are You Literate? Try Again." ; echo "" ; exit ;;
esac
exec startx
```

চালান স্ক্রিপ্টটা। দেখুন, ব্যাশ স্ক্রিনে একটা মেনু তৈরি করে দিচ্ছে। মধ্যে এক এক লাইন ফাঁকা জায়গা ছেড়ে নিচের এই চারটে লাইন, পরপর।

```
1. KDE
2. GNOME
3. FLUXBOX
4. BLACKBOX
```

এবার এই মেনু মিলিয়ে যে নম্বর আপনি টিপবেন সেই রকমের এক্স-উইনডোজ চালু হবে। এর মধ্যে এক্স-উইনডোজ সংক্রান্ত যে জটিলতাটা আছে সেটা ছেড়ে দিন, অন্যটুকুকে ভাবুন। আপনি 'NUMBER' বলে একটা ভ্যারিয়েবল তৈরি করলেন, তার চার চারটে আলাদা মান হতে পারে। '$NUMBER' হতে পারে '1' বা '2' বা '3' বা '4'। এর বাইরেও হতে পারে, সম্পূর্ণ অন্য কিছু, তখন আপনি তাকে মাইল্ড গালি দিয়ে নিচ্ছেন, 'Are You Literate? Try Again.'। এবং এই পাঁচ নম্বর সম্ভাবনাটা ঘটলে স্ক্রিপ্টটা এখানেই শেষ হয়ে যাচ্ছে। আপনাকে আবার './xpick' বলে নতুন করে শুরু করতে হচ্ছে। মানে এই লুপ যাকে 'case' বলে ডাকছে তার এই পাঁচটা সম্ভাবনা। সম্ভাবনাগুলোকে মিলিয়ে লুপটা শুরু 'case $NUMBER in' লাইন থেকে এবং শেষ 'esac' লাইনে। ঠিক 'case' কথাটার উল্টো। ইফ লুপে যেমন দেখেছি আমরা। 'if' লুপের সঙ্গে খুবই মিল 'case' লুপের। এই প্রোগ্রামটা 'if' লুপ দিয়েও করা যেত। 'NUMBER' ভ্যারিয়েবলের প্রতিটি মানের জন্যে একটা করে 'if' শর্ত লিখতে হত, মানে লেখার খাটনিটা একটু বেশি হত।

এবার আমরা তৈরি, আপনার সিস্টেমেই যে স্ক্রিপ্টগুলো দেওয়া আছে, অজস্র, সেগুলো পড়ার কাজে হাত দেওয়ায়। মানে, আপনি এখন আপনার সিস্টেম পড়া এবং শেখা শুরু করতে পারেন, মানে আমার কাজ খতম। পয়লা নভেম্বর থেকে ধরলে, আজ বাইশে মার্চ, মানে একশো বাহাত্তর দিনের আমাদের এই ম্যারাথন খতম। যদিও, যেমন বলেছিলাম, আপনাকে একটু গালাগাল দেওয়ার ব্যাপার আছে, সেটা আসছে ছোট হরফে, প্রতি দিনের শেষেই যেমন এসেছে।

> প্রথমে বলি একটা ভালো কথা। শুধু বইটার জন্যে না, আমার জন্যেও ভালো। আজকাল ভালো কিছু এত কম দেখি, একটু কিছু দেখলেই মনটা ভারি ভরে যায়। এই বইটা লিখে চলাটা আমায় একটা অন্য অনুভব দিয়েছে। আত্মীয়তার অন্য একটা মাত্রা। বইটার জন্যে আমার কী পরিমাণ খাটনি গেছে সেটা আপনারা আন্দাজ করতে পারছেন। সেই খাটনিটা বোধহয় দিতেই পারতাম না এই অভিজ্ঞতাটাকে বাদ দিয়ে। লেখা এবং পড়া — গত বেশ কিছু বছর মোটামুটি ভাবে এটাই আমার কাজ। বই লেখার প্রক্রিয়াটা সেখানে খুবই রোজকার একটা অভিজ্ঞতা, নিজে না হোক অন্যে, আমার চারপাশে নিয়তই লেখা হচ্ছে, এই বই বা ওই বই। কিন্তু এতগুলো বছরের সেই বইঅভিজ্ঞতাতে এরকম কিছু আমি কখনো দেখিনি। কারুর না। না আমার বেলায়, না অন্য-কারুর বেলায়। সেটা এই সামগ্রিকতার অনুভব।
>
> ধরুন এই বইটা লিখেছি শেষ অব্দি আমি, আমার মেশিনে, আমার কিবোর্ডে। আমি যদি বইটার অনেকটা মা হই, বেশ কিছুটা মা সঙ্কর্ষণ, কিছুটা করে মা তথাগত অশেষ এরা। এটা কী করে সম্ভব তাও আমি ভালো বুঝিনি। প্রতি দিন, প্রতিটা দিন, যে প্রক্রিয়ায় সঙ্কর্ষণরা বইটা বিয়োনোর কাজে পার্ট নিয়েছে, সেটা আমি কোনোদিন কারোর বইয়ের বেলাতেই করিনি, করার কথা ভাবিওনা, আসলে আমি ওদের চেয়ে অনেকটা বুড়ো হয়ে গেছি, তাই অনেকটা নীচও।
>
> এবার, যে গালাগালটা আপনাদের দেওয়ার, তার সঙ্গে এর একটা খুব নিকট সম্পর্ক। এই যত্নের প্রক্রিয়ার মধ্যেই সঙ্কর্ষণরা সেই সবই করেছে এই বইয়ের জন্যে যা যা করার কথা, এবং যা যা করার কথা নয়, বোধহয় তাও। সেরকম চেষ্টার ঠেলাতেই হয়ত, একটা সম্ভাবনা দেখা দিয়েছে, ফ্রি সফটওয়ার ফাউন্ডেশন থেকে এই বইটা প্রকাশের, হয়ত, এখনো নিশ্চিত কিছু না, এবং হয়ত, বাংলার পাশাপাশি ইংরিজি মালয়ালম ইত্যাদি অনুবাদেও। কিন্তু বেদনাটা ঠিক এইখানেই। আজ যদি এর অন্য ভাষায় অনুবাদ হয়, ইংরিজি হলে তো বটেই, মলয়ালম হিন্দি তামিল ইত্যাদির বেলাতেও তাই, একটা অপমান তাদের মেনে নিতে হবেনা, এই বইটা লেখার একদম গোড়া থেকেই যেটা ছিল। আজ যদি এই বইটা আমি গোড়া থেকেই ইংরিজিতে লিখতাম তাহলেও যে অপমানটা হতনা।

বইটা লেখা হচ্ছে গ্নু-লিনাক্স নিয়ে, গ্নু-লিনাক্সের অতুলনীয় শক্তিমানতায় অভিভূত হয়ে, অথচ গ্নু-লিনাক্সে কিন্তু লেখা যাচ্ছেনা। অঙ্কুর বাংলা নিয়ে সায়মিন্দু-সঙ্কর্ষণদের প্রচুরতর পরিশ্রমের পরেও, আজো গ্নু-লিনাক্সে, ওপনসোর্স জগতে বাংলায় খুব ভালো কিছু উপায় নেই। যা আছে তাতে এত বড় একটা লেখা শুধু অতুলনীয় রকমের কষ্টসাধ্য নয়, বেশ কিছুটা বোকামিও। এত বেশি নিজেকে অপব্যয় করে চলতে হবে সেইসব উদ্ভট প্রক্রিয়ায়। অথচ, ইংরিজি তো ছেড়েই দিন, তামিল মলয়ালম হিন্দিতে কিন্তু আছে। কেন? আপনার জন্যে। আপনিও আছেন এই বইটায়, আপনার অনড় এবং সংবেদনহীন অনুপস্থিতি নিয়ে, যদি আপনি থাকতেন, আপনি, আপনি, আপনারা প্রত্যেকেই, তাহলে এই বইটা বাংলাতেই লেখা যেত, গ্নু-লিনাক্সেই। ভারতের বেশ কিছু আঞ্চলিক ভাষাতেই গ্নু-লিনাক্সে কাজ করার উপায় যে তৈরি হয়েছে, সেটা আকাশ থেকে পড়েনি। আমার সত্যিই , শুধু খারাপ লাগা নয়, বেশ লজ্জাও লাগছে, আপনার কারণে।

glt-mad@ilug-cal.org

সংকলন ও রচনা : মধ্যমগ্রাম জিএলটি-র (glt-mad@ilug-cal.org) তরফে : ত্রিদিব সেনগুপ্ত