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 দিনে ব্যাশের কাছে তুলে রাখা আগে-ব্যবহৃত আদেশের ইতিহাসের প্রসঙ্গে কনফিগারেশনের এই আলোচনাটাকে আমরা একটু এনেছিলাম। 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, ইউজার প্রোগ্রাম বলতে আমরা কী বুঝছি। বাইনারি এবং সিস্টেম বাইনারির আলোচনায় রুটের সঙ্গে অন্য ইউজারের অধিকারের তফাতের কথা বলেছি। 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। এখানে নেটওয়ার্কের একটা ধারণা উপ্ত রয়ে যাচ্ছে। গ্লোবাল একটা ব্যবস্থা মানে নেটওয়ার্কাবদ্ধ প্রতিটি মেশিনের প্রতিটি ইউজারের বেলাতেই যা সত্যি। আর লোকাল মানে একটা মেশিনে বা একটা ইউজারের বেলায় যা সত্যি। যেখানে আমাদের মত একটা স্ট্যান্ড-অ্যালোন একা-একলা পিসি সিস্টেম নিয়ে আলোচনা হচ্ছে, তখন ওই একটা মেশিনের প্রসঙ্গটা আর থাকছে না, লোকাল বলতে দাঁড়াচ্ছে যা একজন ইউজারের বেলায় সত্যি। একজন ইউজার হিশেবে আমি এই ব্যবস্থা করতেই পারি যে, সত্যিই এটা করা আছে আমার, লগ-ইন করা-মাত্র ব্যাশ আমায় একটা ছোট্ট সাইজের আপত্তিকর বাণী শোনায়, অফেনসিভ ফরচুন। 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 জানে, সেই 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`' দিয়ে শর্ট এবং অফেনসিভ ফরচুন বাণী শোনানোর ব্যবস্থাটা আমার মানে ইউজার 'dd'-র হোম ডিরেক্টরিতে করা আছে ব্যাশের কনফিগারেশনে। কিন্তু এটা ইউজার 'manu' বা 'piu' বা 'atithi'-র নেই। তাই এটা লোকাল সেটিং। এরকম ইউজার প্রোগ্রামেও করা আছে। আমি ইম্যাক্স খুললেই সেটা আমার নিজের কিছু পছন্দ মোতাবেক চলে, সেটাও লোকাল সেটিং। এরকম অনেক প্রোগ্রামেই আছে। ভিম (vim) সফটওয়ারটার একটা নতুন ভার্শন, ভার্শন ৬.২, যেটা সুজে ৮.২-এ দেওয়া ছিলনা, নামিয়েছিলাম নেট থেকে।

ফাইলের নাম 'vim-6.2.tar.bz2'। তাকে '/opt' ডিরেক্টরিতে '`tar xvjf vim-6.2.tar.bz2`' কমান্ড দিয়ে ভেঙে, সদ্য-গজানো '/opt/vim-6.2' ডিরেক্টরিতে '`./configure && make && make install`' করে ইনস্টল করেছিলাম (ইনস্টল করার কায়দাটা অকারণেই একবার বলে নিলাম, গোটাটা আপনাকে মনে পড়ানোর একটা চেষ্টা, কোথায় এটা এসেছিল, মনে করুন)। এই গোটাটাই করেছি রুট হয়ে, কারণ, এটা আমার মেশিনের সুজে ৮.২ সিস্টেমে গ্লোবাল ইনস্টল। এই নতুন ভিমটা এবার আমার সিস্টেমের প্রত্যেক ইউজারই 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 ইউজার 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 অধিকারের দরকার পড়ে যা তার নেই। তার অপ্রশ্নেয় এলাকা তার হোম ডিরেক্টরিটুকু। এমনকি সেখানেও রুট যে কোনো বদল যে কোনো সময়েই ঘটাতে পারে।

ইউজার বা সিস্টেম দুরকম প্রোগ্রামই চালানোর মুহূর্তে নিজেদের কনফিগারেশন ফাইল একবার পড়ে নেয়। কোনো কোনো সিস্টেম প্রোগ্রাম চালু হয় একদম বুটের সময়ে, তাই তাদের কনফিগারেশন ফাইলও পড়া হয় ওই সময়েই। '/etc' ডিরেক্টরির সেসব 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, ইউজার যদি নিজের মত কোনো বদল আনতে চায় সেটা ঘটাবে তার নিজের হোম ডিরেক্টরিতে রাখা বাড়তি একটা ব্যক্তিগত কনফিগারেশন ফাইলে। এই ফাইল মোতাবেক বদলটা ঘটবে শুধু ওই প্রোগ্রামের ওই ইউজারের ব্যক্তিগত বা লোকাল ব্যবহারেই। প্রোগ্রামটার গ্লোবাল ব্যবহার নিয়ন্ত্রিত হবে '/etc' ডিরেক্টরির গ্লোবাল কনফিগারেশন দিয়ে। আর যদি প্রোগ্রামটা গ্লোবালি ইনস্টল না-করে, শুধু একক ইউজারের জন্যে লোকালি করা হয়ে থাকে, তার হোম ডিরেক্টরিতেই, 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 ইউজার 'dd'-র হোম ডিরেক্টরিতে বা '/home/dd'-তে আছে '.gnu-emacs' বা '.emacs'। যা মূলত '/etc/skel' ডিরেক্টরির ওই গ্লোবাল কনফিগারেশন ফাইলদুটোরই কপি। পিউ ইম্যাক্সে সি-প্রোগ্রাম লেখা বা কম্পাইল করার জন্যে কিছু সুবিধেজনক বদল এনেছিল, যা লিখে নিয়েছিল ওর হোমে '/home/piu/.emacs' ফাইলে। বদলটা আমার হোমে '/home/dd/.emacs' ফাইলে না-থাকায় সেগুলো আমার ইম্যাক্সে ঘটেনা। এবার ভাবুন, ইম্যাক্স চলার রকমটা। যেই কেউ '`emacs`' মন্তর পড়ে প্রোগ্রামটা জাগিয়ে তুলছে, ইম্যাক্স পড়ে নিচ্ছে '~' ডিরেক্টরি থেকে ব্যক্তিগত কনফিগারেশন মানে '~/.emacs' ফাইল। '~' চিহ্নটা দিয়ে ব্যাশ কোনো ইউজারের হোমকে চিনে নেয়। প্রোগ্রামটা যখন আমি মানে ইউজার 'dd' চালাচ্ছে '~' তখন '/home/dd'। যখন পিউ চালাচ্ছে, '~' তখন '/home/piu'। কোনো ব্যক্তিগত '~/.emacs' ফাইল না-পেলে ইম্যাক্স তখন কাজে লাগায় শুধু '/etc/skel' ডিরেক্টরির গ্লোবাল কনফিগারেশন।

আমার প্রিয় প্রোগ্রাম 'mplayer'। ম্যাজিক মাউন্টেনের ওই আনুভূমিক আর উল্লম্বের রসিকতাটাকে একটু বদলে নিয়ে বলাই যায়, যখনই অক্ষর দেখছি-না, মানে লিখছি বা পড়ছি না, তখনই 'mplayer'-এ ফিল্ম দেখছি। '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' বা গ্লোবাল কনফিগারেশন ফাইলেরই একটা কপি, তারপরে তাতে নিজের ইচ্ছে এবং পছন্দ মাফিক বদলগুলো ঘটালেন। সিস্টেমে যদি দুটো ফাইলই থাকে, এবং দুটোর সবকিছু হুবহু না মেলে, 'mplayer' তখন মানবে '~/.mplayer/config' ফাইলে দেওয়া নিদানটাই। শুধু এই দুটো নয়, কনফিগারেশন খোঁজার একটা তৃতীয় জায়গাও আছে, তিনটেই খোঁজে 'mplayer', পরপর। তৃতীয় জায়গাটা '/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, 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 ইমোটিকনের টেক্সট প্রতিরূপটাও দেওয়া আছে, '`:)`'। আর এরপরেই সেই অনবদ্য শেষ লাইন, যেখানে সেই পথটা লিখে দেখিয়ে দেওয়া আছে। আরটিএফএম মানে রিড-দি-ফাকিং-ম্যানুয়াল শব্দবন্ধটা 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...
```

ইউজারের স্তরে যে কোনো প্রোগ্রামই মোটামুটি প্রথমে খোঁজে '/etc' ডিরেক্টরি, সেখান থেকে ডিফল্ট কনফিগারেশন পড়ে নেয়। তারপর দেখে নেয়, ইউজার সেই প্রোগ্রামের কোনো কাস্টমাইজেশন বা নিজের পছন্দ অনুযায়ী বদল ঘটিয়েছে কিনা। সেটা দেখে ইউজারের হোমে রাখা ডটানন ফাইলে বা আরসি ফাইলে। 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। চার নম্বর দিনে, ১৯৬২-তে এমআইটির সিটিএসএস বা কম্পেটিবল টাইম শেয়ারিং সিস্টেমের কথা বলেছিলাম মনে আছে? ওই সিস্টেম বুট করার সময় ব্যবহৃত হত রানকম (runcom) নামের ফাইল। সেই রানকম থেকে এসেছে আরসি কথাটা। লাগে এটা জানানোর পরে 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 বা চালু-হওয়া-কালীন ব্যবহৃত স্ক্রিপ্টের জন্যেই, একটা কমন-নাউন হিশেবে।

ইউজারের স্তরে এই ডট ফাইল এবং আরসি ফাইলের মূল প্রয়োজনীয়তাটা কিন্তু রুট এবং ইউজারের অধিকারের পার্থক্য। নয় নম্বর দিনে আপনারা দেখেছেন, '/etc' ডিরেক্টরির বিভিন্ন কনফিগারেশন ফাইল বদলে ফেলাটা শুধু শ্রমসাধ্যই নয়, যথেষ্ট গা-ছমছমেও বটে, কিছু যদি ঘেঁটে যায়? সিস্টেমে প্রতিটা কিছুই অন্য কিছুর সঙ্গে অতিনির্ণয়ের বা ওভারডিটারমিনেশনের সম্পর্কে আছে, প্রতিটি কিছুই অন্য প্রতিটি কিছুর দ্বারা নির্ণীত এবং নির্মিত। তার মানে একটা কোনো ভুল বদল কতদূর অব্দি কোনটাকে ঘেঁটে দিতে পারে এর একটা আন্দাজ আগে তৈরি হওয়ারও দরকার পড়ে। কোনো প্রোগ্রামের কিছু বদলাতে চাইলে, সহজ হল নিজের হোমে রাখা ওই প্রোগ্রামের কনফিগারেশন ফাইলটা বদলে নেওয়া। বদলাতে গিয়ে ভাবে ঘেঁটে গেলে, স্রেফ '/etc' ডিরেক্টরির কনফিগারেশনটা ফের কপি করে হোমে বদলানো ফাইলের জায়গায় বসিয়ে দাও, ব্যাস। আর, ব্যক্তিগত সিস্টেম না-হলে '/etc' ডিরেক্টরির কোনো ফাইলে হাত দিতে পারেনা ইউজার। ওটা গ্লোবাল, তাই রুটের এলাকা। তাই এই গ্লোবাল লোকাল দুই সেট ফাইল। গ্লোবাল কনফিগারেশন '/etc' ডিরেক্টরিতে। লোকাল কনফিগারেশন ইউজারের হোমে, ডট ফাইল বা আরসি ফাইল। এখানের উদাহরণগুলোর সঙ্গে মিলিয়ে এবার কনফিগারেশন ফাইলগুলো চিনে নিতে পারবেন নিজের হোমে। অনেক ফাইলই দেখবেন এক্স-উইনডোজ সংক্রান্ত, মানে, আমাদের এই পাঠমালার এলাকার বাইরে। ও, কী ভাগ্যি, তখন বাদ দিয়েছিলাম। অবশ্য আমি খুব একটা পারতামও না বুঝিয়ে বলতে, গুইটা নিজেই তো আন্দাজে-মান্দাজে করি। বেশিরভাগই। দু-একটা কাজ অবশ্য খুব ইন্টারেস্টিং লাগে, বলেছি আগেই, যেমন এক্সউইনডোজের কনফিগারেশন ফাইল 'XF86Config'-টা বদলানো। ফাইলটা থাকে '/etc/X11' ডিরেক্টরিতে। নয় 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। এবার ব্যাশ শেলের কনফিগারেশন ফাইল। কনফিগারেশন ফাইলের বৃত্তান্ত লিখতে যে লেখাগুলোর সাহায্য পেয়েছি তাদের রেফারেন্স দিয়ে রাখি। মূল রেফারেন্স অবশ্যই নিজের মেশিনের সুজে সিস্টেম। অন্যগুলো সবই নেট থেকে, আমার কাছে নামানোই আছে, যদি

কারুর লাগে, আমারই মত যাদের নেট করতে করতে খরচ গুণতে হয়, মেল করুন, কিছু একটা করা যাবে। পাঠমালাটা লিখতে গিয়ে নেট থেকে যা কিছু কাজে লাগিয়েছি, সবই রাখা আছে হার্ডডিস্কে।

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 'bash' মানে বর্ন-এগেন-শেল (**B**ourne-**A**gain-**SH**ell)। বেল ল্যাবরেটরির ইউনিক্স রিসার্চ ভার্শন সেভেনের জন্যে বর্ন শেল লিখেছিলেন স্টিভ বর্ন। তার পর আরো শেল লেখা হয়েছে। কর্ন-শেল, সি-শেল ইত্যাদি। বর্ন শেল যা যা পারত তা সবই পারে ব্যাশ, এবং আরো কিছু পারে। বর্ন শেলের স্ক্রিপ্ট ব্যাশেও চালানো যায়। গ্নু-লিনাক্সে ডিফল্ট শেল ব্যাশ, সিস্টেম এবং ইউজারের মধ্যে দোভাষির কাজ করে। ইউজারের আদেশ পেয়ে 'bash' জেগে ওঠে, সেই আদেশ পাঠিয়ে দেয় কারনেলের কাছে, তারপর কাজ সমাপ্ত হলে তার ফলাফল ইউজারের কাছে পৌঁছে দিয়ে ফের ঘুমোতে চলে যায়। এর পর ব্যাশকে অনেক বিশদে বুঝব, এখন ব্যাশ শেলের কনফিগারেশন।

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, পরের তিনটে ইউজারের লোকাল ফাইল। এর সবকটা ফাইলই যে সবসময় থাকবে তা নয়, কিন্তু চাইলে এদের যে কোনোটাই ব্যবহার করা যায়। সুজে সিস্টেমে আমার হোমে ডটানন ফাইলের তালিকার মধ্যে ব্যাশ বিষয়ে ফাইল '~/.bash_history', '~/.bashrc' এবং '~/.profile'। শেষটা মানে '~/.profile' হল ওই '~/.bash_profile' ফাইলেরই নামান্তর। প্রথমটা, '~/.bash_history', সাত নম্বর দিনে আমরা চিনেছিলাম, এতে আগে ব্যবহৃত আদেশের তালিকাটা তোলা থাকে। কোনো '~/.bash_logout' ফাইল সুজেতে থাকেই না, তবে চাইলে কোনো ইউজার বানিয়ে নিতেই পারে। সুজেতে আমার হোম মানে '/home/dd' থেকে '.bashrc' আর '.profile' এই ডটানন ফাইলদুটো তোলা যাক। 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। ওটা আমার অবদান। নিজের অবদান মাত্র একবার ঘোষণা করে সুখ কই? শুধু ফরচুনের অপশনটা খেয়াল করুন। '.bashrc' ফাইলে '/usr/bin/fortune'-এর অপশান '`os`', আর '.profile' ফাইলে '`s`'। লগ-ইনের সময়ে '.profile' পড়ে ব্যাশ শুধু একটা ছোট্ট ফরচুন শোনায়, '`s`' মানে শর্ট বা ছোট। আর 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 এবং ছোট্ট বাণী, 'os'। 'o' মানে অফেনসিভ। কিন্তু এর মানে এই নয় যে '~/.bashrc' বা '~/.profile' ফাইলে শুধু এই-ই থাকবে, এছাড়া আরো বহু কিছু থাকতে পারে এবং থাকেও, ইউজারের ইচ্ছে অনুযায়ী। কী কী ভাবে কী আনা যায় সেখানে সেগুলোর বিশদ আলোচনায় আসছি। আমি এখানে এদুটোকে তুলে জাস্ট আপনাদের একটু অভ্যস্ত করলাম, আর এদের মধ্যে রাখা ওই কমেন্ট লাইনগুলো পড়াতেও চাইছিলাম। আমার মেশিনের সুজে সিস্টেমে কোনো '~/.bash_logout' নেই, যদিও চাইলেই রাখা যেত। স্ল্যাকওয়ারে ইউজার 'dd'-র হোম থেকে '.bash_logout' ফাইলটা তুলে দিই। মাত্র দু-লাইন।

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

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 ইউজারের নিজের 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। অ্যালিয়াসের কথায় আমরা একটু বাদেই আসছি।

৩।। ব্যাশের গ্লোবাল কনফিগারেশন

যে পাঁচটা ফাইল 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 গ্লোবাল কনফিগারেশন, মানে, এই সিস্টেমে যারা যারা ব্যাশ ব্যবহার করে তাদের সবার ব্যাশকে নিয়ন্ত্রণ করবে এই ফাইলদুটো। গ্লোবাল কনফিগারেশন, তাই '/etc' ডিরেক্টরিতেই পাওয়ার কথা। পরের তিনটে লোকাল কনফিগারেশন, মানে, একজন ব্যক্তি ইউজার নিজের ব্যাশ-ব্যাবহারকে কনফিগার করে এদের দিয়ে। নিজের হোম ডিরেক্টরিতে গোপন বা হিড্ন ফাইল হিশেবে থাকে এরা, কারণ এরা ডটানন, মানে মুখে ডট আছে এদের। শুধু 'ls' করলে দেখা যায়না, 'ls -a' কমান্ড দিলে, অল বা সব ফাইল দেখাতে বললে তখন দেখা যায়। যদি এদের সব ফাইলকে আপনি আপনার হোম ডিরেক্টরিতে না পান, নিজের ফাইল নিজেই লিখে বানিয়ে বা বদলে নিতে পারেন।

আপনার করা বদলে ব্যাশ ঘেঁটে গেলে তাকে ঠিক করে নেওয়ার জন্যে '/etc/profile' তো আছেই — ব্যাশের গ্লোবাল কনফিগারেশন। সাত নম্বর দিনে এনভায়রনমেন্ট ভ্যারিয়েবল বা গেরস্থালির চলরাশির কথা উল্লেখ করেছিলাম, মনে করে দেখুন, 'set' কমান্ড থেকে পাওয়া ফলাফলকে 'less' করে পড়ে পড়ে। সেই চলগুলোকে নিয়ন্ত্রণ করে '/etc/profile'। 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, সেই প্রোগ্রামকে '/etc/profile' জানিয়ে দেয়, ব্যাশের তথা সিস্টেমের ঘর-গেরস্থালির আকার-আকৃতি মানে এনভায়রনমেন্ট ভ্যারিয়েবলগুলো কী হবে। আপনারা যারা উইনডোজ এমিগ্রি, মানে উইনডোজ থেকে গ্নু-লিনাক্সে অভিবাসী, তারা মনে করতে পারবেন, 'c:\autoexec.bat' ফাইল দিয়ে কিছু কাজ করা হত। প্রম্পটের আকার বদলানো, কাজ করার পথে কোন কোন ডিরেক্টরি থাকবে সেই নির্দেশগুলো বদলানো, ইত্যাদি। এই কাজগুলো সবই এবং আরো বহু বহু কিছু করা যায় '/etc/profile' দিয়ে। '/etc/bashrc' ব্যাশের আর একটা গ্লোবাল কনফিগারেশন ফাইল। কী কী অ্যালিয়াস এবং কী কী ফাংশন ব্যবহার করা যাবে, বা ব্যাশ চালু হওয়ার সময় কী কী স্ক্রিপ্ট চালিয়ে নেবে, তার হদিশগুলো থাকে এই '/etc/bashrc' ফাইলে। অ্যালিয়াস মানে শর্টকাট, ব্যাশকে অল্প কথায় বেশি আদেশ দেওয়ার উপায়, অ্যালিয়াস নিয়ে আলাদা আলোচনায় আসছি আমরা, পরের সেকশনেই। অ্যালিয়াস ছাড়া এই গ্লোবাল কনফিগারেশন ফাইল '/etc/bashrc'-এ থাকে ফাংশনদের তালিকা, ব্যাশ শুরু হওয়ার সময়েই যাদের চালু

করে নেবে। এই ফাংশনগুলো ধরতে পারেন ছোট ছোট স্ক্রিপ্ট, ছোট ছোট কাজের জন্যে। কোনো ব্যাশ প্রোগ্রামে বা ব্যাশ কমান্ড প্রম্পটে ফাংশনগুলো দিলে ব্যাশ ওই ছোট ছোট কাজগুলো করে দেয়। ফাংশনের কথায় আসছি। '/etc/bashrc' না থাকলে এসব দেওয়া থাকে '/etc/profile'-এ। ডিস্ট্রো থেকে ডিস্ট্রোয় এদের নাম একটু ভিন্ন হতে পারে, সুজেতে '/etc/bash.bashrc' নামে থাকে। আমার মেশিনের সুজে সিস্টেমে '/etc/profile'-এর সাইজ ২৮৫ লাইন আর '/etc/bash.bashrc' ২৫৩ লাইন।

৪।। অ্যালিয়াস বা ওরফ

এই অ্যালিয়াস বা ওরফ মানে শর্টকাট। ফাটা-রতন বা কাটা-গমা বললেই পুলিশ চিনে যায় এর পিছনে পিতৃদত্ত নামে কমল চন্দ্র সাহা বা পরেশ মাইতিকে, মানে এলাকার কোনো Sনামধন্য Sমাজসেবীকে, এবং সে কোনো নবতর সমাজসেবায় হাত দেওয়া মাত্রই মাসোহারার রেট বাড়ায়। ওরফ বা অ্যালিয়াসটার পিছনে মূল ঘুঘুটাকে চেনে পুলিশ, তার পিছনে তার দাদা-কাম-পিতাকেও, মাসোহারার কারণেই চিনে রাখতে হয়। তেমনি, ঠিক মাসোহারা নয়, কমান্ড প্রম্পটে কাজকর্মের সুবিধের স্বার্থে সিস্টেমে ওরফ বা অ্যালিয়াস বানিয়ে নেওয়া যায়। কিছু অ্যালিয়াস ডিস্ট্রোর তরফেই বানিয়ে দেওয়া থাকে সিস্টেমে। মেশিনে মোট কী কী অ্যালিয়াস ডিস্ট্রো থেকেই করা আছে সেটা জানার উপায় অ্যালিয়াসের ডকুমেন্টেশন, কিন্তু অ্যালিয়াসের নিজের কোনো ম্যানপেজ নেই। অ্যালিয়াস হল ব্যাশের একটা আভ্যন্তরীন কাজ, বিল্ট-ইন ফাংশন। ব্যাশের ম্যানপেজটা বা নয় নম্বর দিন 'changeman' ব্যাশস্ক্রিপ্ট দিয়ে ওয়েবপেজে যে ম্যানসমগ্রটা বানিয়েছিলাম, তার 'bash.1.html'-টা খুলুন। বেশ 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 সিস্টেমের গোটা অ্যালিয়াসের তালিকাটা দেখাবে। এবার আমার মেশিনের সুজে সিস্টেমের এই তালিকাটাকে রিডাইরেক্ট করে একটা ফাইলে এনে তার থেকে কয়েকটা 'ls' সংক্রান্ত কয়েকটা অ্যালিয়াস তুলে দিলাম, আছে কিন্তু প্রচুর।

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

এর মধ্যে আমাদের খুব পরিচিত 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' সেরকমই বলে দিয়েছে।

নিজেও বানিয়ে নিতে পারেন অ্যালিয়াস। ধরুন আপনি চাইছেন, নতুন প্রবন্ধে হাত দেওয়ামাত্র ব্যাকগ্রাউন্ডে গান বাজতে শুরু করবে, গান না-শুনে লেখা যায়? আর গানগুলো আছে আপনার হোমে, 'music' বলে একটা ডিরেক্টরিতে, সব 'mp3' ফাইল। 'mp3' মানে সাধারণ অডিও বা শব্দতথ্যের কোঁকড়ানো ছোটকরা একটা আকার। আর প্রবন্ধগুলোও রয়েছে হোমের ভিতরেই, 'probondho' ডিরেক্টরিতে। এবার এরকম একটা অ্যালিয়াস বানাতে চান যা প্রবন্ধ লেখার ডিরেক্টরিতে নতুন প্রবন্ধ খুলে দেবে ইম্যাক্সে এবং একই সঙ্গে গান বাজাতে শুরু করবে। কমান্ড দিন, 'alias likhi='mpg123 ~/music/*.mp3 & cd ~/probondho;emacs'' এবং এন্টার মারুন। সত্যিই তৈরি হল কিনা সেটা দেখে নিতে আর একবার 'alias' কমান্ডটা মেরে দেখে নিন, আগেরগুলোর সঙ্গে এই নতুন অ্যালিয়াসটা যোগ হয়েছে কিনা। এখন থেকে 'likhi' কমান্ড দিলেই ব্যাশ এনে দেবে '~/probondho' ডিরেক্টরিতে, ইম্যাক্স খুলে দেবে, এবং ব্যাকগ্রাউন্ডে বাজতে থাকবে '~/music' ডিরেক্টরির '*.mp3' ফাইলগুলো একের পর এক। গানটা কিন্তু সত্যিই ব্যাকগ্রাউন্ডে চলে গেছে। আমাদের দেওয়া আদেশের মধ্যে '&' চিহ্নটা থাকায় ব্যাশ এটা করেছে। যে কোনো ক্রিয়াকে '&' নেপথ্যে পর্দার আড়ালে নিয়ে যায়। লিখতে লিখতে গানটা যদি বন্ধ করতে চাইলে সরাসরি হবেনা। প্রথমে চলন্ত ইম্যাক্সকে নেপথ্যে নিতে হবে, 'Ctrl-Z' সুইচ টিপে। ইম্যাক্স হাওয়া হয়ে কমান্ড প্রম্পট ফেরত আসবে। ইম্যাক্স কিন্তু মরে যায়নি। এখনি 'fg' লিখে এন্টার মারলেই ইম্যাক্স ফেরত চলে আসবে স্ক্রিনে। ইম্যাক্স ব্যাকগ্রাউন্ডে চলে গেছে, গান বাজানোর 'mpg123' প্রোগ্রামের

মত। ইম্যাক্সকে যেভাবে একবার স্রেফ 'fg' সেধেই ফেরত আনতে পারলাম আমরা, 'mpg123' তা হবেনা। একেও ফোরগ্রাউন্ডে আনতে হবে 'fg' মেরেই, কিন্তু, এখানে একটু মৃদু জাটিল্য আছে। এখন ব্যাকগ্রাউন্ডে আর একটা নয়, দুটো জব আছে। '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 করে গাইতে পারলে সিস্টেম কখনো কখনো রিঅ্যাক্ট করে বলেই শুনেছি, কিন্তু আমরা যারা 'সুরের গুরু দাও গো সুরের দীক্ষা' গাইলে অহল্যা জেগে ওঠে পাষাণ ফুঁড়ে, ট্রাফিক থেমে যায়, এবং নিকটতম অ্যামওয়েজ বিক্রেতা নাগা-সন্ন্যাসের সংকল্পে হিমালয়ের টিকিট কাটে, তাদের কী হবে?

তাদের জন্যে আছে 'jobs'। কমান্ডটা দিন, 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, 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', পরপর, 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 2' কমান্ড দিলে ফোরগ্রাউন্ডে আসবে ইম্যাক্স, এবং 'fg 1' কমান্ড দিলে ফোরগ্রাউন্ডে আসবে 'mpg123'। এদের মধ্যে ইম্যাক্সকে টাটা জানানোর উপায় তো জানেনই, 'Ctrl-X,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? 'pwd' কমান্ডটা মনে আছে? বারবার ব্যবহার করেছি আমরা, কোন ডিরেক্টরিতে আছি এবং কাজ করছি সেটা জানতে। সেখান থেকে মনে করুন তো। 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। 'alias' যেমন ব্যাশের বিল্ট-ইন কমান্ড, 'jobs' বা কাজের তালিকা দেখানোটাও তাই।

এইমাত্র যে অ্যালিয়াসটা আপনি বানালেন সসঙ্গীত সাহিত্যচর্চার, সেটা কিন্তু অস্থায়ী অ্যালিয়াস। পুলিশের রেকর্ডে এখনো তার নাম তোলা হয়নি। পরেরবার লগ-ইন করে আর পাবেন না অ্যালিয়াসটা। পরের বার পাবেন কী করে? এর উপায়, কোনো একটা কনফিগারেশন ফাইলে যোগ করে দেওয়া। গ্লোবাল নয়, লোকাল কনফিগারেশনে। সবার এই শিল্পমননের দরকার পড়বে না। সেই অধিকারও নেই আপনার, রুট না-হলে। নিজের জন্যে তুলে রাখতেই পারেন। হোম ডিরেক্টরির ডটানন ফাইল '~/.bashrc' ইম্যাক্স দিয়ে খুলে তার একদম শেষে হুবহু অ্যালিয়াস বানানোর কমান্ডের গোটা লাইনটা লিখে দিন। অমন কোনো ফাইল আপনার হোমে না-থাকলে, আলতো করে ছুঁয়ে দিন, টাচ কমান্ডটা মনে আছে? 'touch ~/.bashrc'। এবার তাকে ইম্যাক্স দিয়ে খুলে টাইপ করে নিন। 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? যদি ফাইল থাকে তো তার শেষে লাগিয়ে দেবে,

আর না-থাকে তো বানিয়ে লাগিয়ে দেবে। এতে সুবিধেটা এই যে ওই নামে ফাইল থাকলেও সেটা উড়িয়ে দেবেনা, এমনিতে ক্যাট যা করে। এখন থেকে প্রতিবার লগ-ইন করেই অ্যালিয়াসটা পাবেন। তবে, '~/.bashrc' ফাইল 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 — "ব্যাশিতে শেখেনা কেহ না-ঘেঁটে মেশিন"।

৫।। ব্যাশ এবং তার ভ্যারিয়েবল

ব্যাশের কনফিগারেশন ফাইল আলোচনা করতে গিয়ে বারবার 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, সেই জন্যেই এদের বলে চল বা চলরাশি, ভ্যারিয়েবল, কারণ এরা ভ্যারি করে। চল রাশির 'রাশি'-টা আসে গণিতে ব্যবহারের সূত্রে। শূন্য এক আর দুই নম্বর দিনে আমরা বারবার স্মৃতিভূমির কথা এনেছি, নানা ধরনের নানা আকারের স্মৃতিভূমি। কখনো হার্ডডিস্কের স্মৃতির ভাঁড়ার, কখনো র‍্যামের উদ্বায়ী স্মৃতি। পরে আট নম্বর দিনে ভৌতিক স্মৃতি বা ভারচুয়াল মেমরির কথাও বলেছি, র‍্যামকে দিয়ে গোটা কাজ করতে গিয়ে অপারেটিং সিস্টেমের কারনেলের কাছে স্মৃতি রসদের যাতে টান না-পড়ে, সেই জন্যে, ১২৮ বা ২৫৬ বা ৫১২ যত এমবি ভৌত র‍্যামই থাক, তার সঙ্গে সঙ্গে পোঁ-ধরার মত একটা বাড়তি রসদ, সোয়াপ ফাইল, বা হার্ডডিস্কের অস্থায়ী ভাবে লিখে রাখার জায়গা। এরা সবাই স্মৃতি।

আমরা জানি, 'dd' নামের ইউজার হয়ে '`echo $PATH`' মারলে যে পথনির্দেশটা পাই সেটা হল 'dd'-র দেওয়া আদেশের প্রোগ্রামগুলোকে ব্যাশ কোথায় কোথায় খুঁজবে সেই ডিরেক্টরিগুলোর তালিকা। আবার 'root' হয়ে যখন একই কমান্ড দিয়ে পাই আলাদা একটা পথনির্দেশ। এর মানে, এদের প্রত্যেকের জন্যে সিস্টেমের স্মৃতিভূমিতে ব্যাশ 'PATH' নামের এক এক ফালি জমি তুলে রেখেছে। প্রত্যেকের আলাদা আলাদা জমি। যখনই '$PATH' ইকো করতে বলা হচ্ছে, সেই জমিতে যা ফলে আছে সেই ফলনটাকে ব্যাশ ইকো বা প্রতিধ্বনি করে দিচ্ছে স্ক্রিনে। এই জমিটাই হল 'PATH' নামের একটা ভ্যারিয়েবল। আর ফলনটা তার মান বা ভ্যালু, আমরা জানি '$' দিয়ে ভ্যালুকে বোঝানো হয়। তাই '$PATH' হল 'PATH' নামক ভ্যারিয়েবলের ভ্যালু। কোনো ইউজার, বা রুট, বা ইউজার থেকে '`su`' করে রুট, যেই কেউ লগ-ইন করছে, তখনই আলাদা একটা ব্যাশ প্রসেস চালু হচ্ছে। মাল্টিপ্লেক্সিং-এর আলোচনা মনে করুন, কোনো প্রোগ্রাম ফাইলকে গ্নু-লিনাক্সে সরাসরি চালানো হয়না, তার একটা প্রতিরূপকে তুলে নেওয়া হয় ভৌতিক স্মৃতিতে। ভৌতিক স্মৃতিভূমি থেকে চলতে থাকা সেই প্রোগ্রামটাই তখন একটা প্রসেস। যখনই কেউ লগ-ইন করছে, বা ইন্টারাক্টিভলি মানে জ্যান্ত দেওয়া-নেওয়ার রকমে ব্যাশকে চালাচ্ছে, সিস্টেম ব্যাশ নামক প্রোগ্রাম ফাইলের একটা প্রতিরূপ নিজের স্মৃতিতে তুলে নিচ্ছে। জন্ম নিচ্ছে আর একটা ব্যাশ প্রসেস। খুব সুন্দর করে এই প্রসেসের ব্যাপারটা বোঝানো আছে ট্যানেনবমের 'মডার্ন অপারেটিং সিস্টেমস' বইটাতে, আর গ্নু-লিনাক্সের মত করে এই ব্যাপারটা প্রাথমিকভাবে বোঝার জন্যে আছে এরিক রেমন্ডস-এর 'ইউনিক্স অ্যান্ড ইন্টারনেট ফান্ডামেন্টালস হাউ-টু', হাউ-টুগুলোর মধ্যেই পাবেন।

যখনই একটা নতুন ব্যাশ-প্রক্রিয়া চালু হচ্ছে, ইউজার, রুট, ইউজার থেকে '`su`' করে রুট, যার জন্যেই হোক, চালু হওয়ার আগে ব্যাশ ওই কনফিগারেশন ফাইলগুলো পড়ে নিচ্ছে, এবং সেই কনফিগারেশন ফাইলে ঘোষিত প্রতিটা এনভায়রনমেন্ট ভ্যারিয়েবলের জন্যে একটা করে স্মৃতিখণ্ড বরাদ্দ করছে। এবার যখনই আমরা সেই ভ্যারিয়েবলকে ইকো করতে বলছি, ব্যাশ সেই স্মৃতিখণ্ডে ভরে রাখা মানটাকে স্ক্রিনে শো করে দিচ্ছে। শুধু এই প্রতিবেশ-চল বা এনভায়র্নমেন্ট ভ্যারিয়েবল নয়, আরো এক রকমের চল বা ভ্যারিয়েবল হয়, তাকে বলে লোকাল বা আঞ্চলিক ভ্যারিয়েবল। গ্লোবাল কনফিগারেশন পড়ে যেমন এনভায়র্নমেন্ট ভ্যারিয়েবলদের জন্যে স্মৃতিখণ্ড বরাদ্দ করে ব্যাশ, এই লোকাল ভ্যারিয়েবলদের জন্যে জমি বরাদ্দ করে লোকাল কনফিগারেশন ফাইল পড়ে, যেগুলো থাকে ইউজারের

হোমে। লোকাল ভ্যারিয়েবলগুলোকে বানিয়ে তোলা হয় একজন ব্যক্তি ইউজারের ইচ্ছে অনুযায়ী। বানিয়ে তোলাটা ঘটতে পারে দুই ভাবে। এক, ব্যক্তি ইউজার সরাসরি বানাল, জ্যান্ত রকমে কমান্ড প্রম্পটে কমান্ড দিয়ে। দুই, ব্যাশ চালু হওয়ার সময় যে লোকাল কনফিগারেশন ফাইল পড়ে নেয়, ব্যক্তি ইউজারের হোম ডিরেক্টরি থেকে, সেই ফাইলগুলো মারফত।

সরাসরি বা ইন্টারাকটিভলি মানে 'bash' কমান্ড দিয়েও ব্যাশ চালু করা যায়। তখন তার মধ্যে কোনো একটা ভ্যারিয়েবল বানিয়ে গুঁজে দেওয়া যায়। এই ইন্টারাক্টিভ ব্যাশ প্রসেস তখন ওই ভ্যারিয়েবলের নামে একটা জমি তৈরি করে, যেখানে ইউজারের গুঁজে দেওয়া মানটা রেখে দেওয়া হয়। এই রাখাটা অত্যন্ত সাময়িক। 'exit' কমান্ড দিয়ে বা 'Ctrl-D' করে ব্যাশ থেকে বেরিয়ে যাওয়া মানে এই ভ্যারিয়েবলটারও ইন্তেকাল হয়ে গেল। একটা জিনিষ খেয়াল করুন। বেরিয়ে তো গেলেন এই জ্যান্ত ব্যাশ প্রসেস থেকে। কিন্তু বেরিয়ে গেলেন কোন চুলোয়? কেন? ব্যাশের বাইরে? না, সরি, সেটা যাওয়ার উপায় নেই, যাওয়া মানেও ব্যাশ, আসা মানেও ব্যাশ। তু হাঁ-কর ইয়া না-কর তু হ্যায় মেরি কিকিকিরণ। 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 সিস্টেম, বা কমান্ড প্রম্পটে 'bash' কমান্ড দিয়ে ইউজার নিজে জ্যান্ত রকমে, তার সবই 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। এরও 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। 'মডার্ন অপারেটিং সিস্টেমস' স্তরের টেক্সটবই আমি আমার জীবনে আর দুটো মাত্র দেখেছি — স্যামুয়েলসন-নর্ডহউসের ইকনমিক্স এবং কুরান্টের ক্যালকুলাস। সাসপেন্স থ্রিলারের মত উত্তেজক, টেক্সটবই লিখনা হ্যায় তো অ্যায়সা। একজন পাঠক তথা ছাত্রের গোটা চিন্তাপ্রক্রিয়াটাকেই এতোলবেতোল করে দেওয়া। তবে বইটাকে পড়া দরকার গ্নু-লিনাক্সে নিজের সিস্টেমের সঙ্গে মিলিয়ে মিলিয়ে। ওখানে তো ইউনিক্স উইনডোজ ওএসটু সোলারিস সবই আছে, নৈর্ব্যক্তিক একটা রকমে, প্রতিমুহূর্তে সব ধরনের অপারেটিং সিস্টেমের উদাহরণ টেনে টেনে।

যা বলছিলাম, লোকাল ভ্যারিয়েবল, যাদের সরাসরি, ইন্টারাক্টিভলি, একজন ইউজার বানিয়ে তুলছেন। অ্যালিয়াস দিয়ে যে কাজটা করছিলাম, এবারে ব্যাশের লোকাল ভ্যারিয়েবল দিয়ে সেই কাজটা করা যাক। 'likhi' অ্যালিয়াসটার জন্যে সমান চিহ্নের ডানদিকে যে অংশটা ছিল, একটা ভ্যারিয়েবল বানিয়ে সেই অংশটাকে তার মান 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, ব্যাশ ভ্যারিয়েবলের মান হিশেবে নিয়ে নিত প্রথম স্পেস চিহ্নের আগে অব্দি। এই যে লোকাল ভ্যারিয়েবল 'lekho' তৈরি করলাম, এর মান কত? এবং সেই মানটাকে ব্যাশ কী নামে চেনে? 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 ব্যাশ-নাম। মূল মজাটা অন্যত্র। কমান্ড প্রম্পটে নিছক '$lekho' টাইপ করে এন্টার মারুন। আগে অ্যালিয়াস দিয়ে যে কাজটা হচ্ছিল, সেই কাজটাই হচ্ছে। 'lekho' নামের লোকাল ভ্যারিয়েবল সেই কাজটাই করল, যেটা একটু আগে 'likhi' নামের অ্যালিয়াস দিয়ে করছিলাম।

এই লোকাল ভ্যারিয়েবল 'lekho' 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। 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 ব্যাশ প্রসেস। এখানে আপনি '$lekho' কমান্ড দিয়ে এন্টার মারুন। কী পেলেন? ব্যাশের অজ্ঞতা : কমান্ড নট ফাউন্ড। ব্যাশ এই কমান্ডটা জানেই না। বেচারা জানবেই বা কী করে? আপনি তো আগের ব্যাশ প্রসেস থেকে রপ্তানি করেননি। এই ব্যাশের-মধ্যে-ব্যাশ থেকে বেরিয়ে যান। তার জন্যে 'exit' বা 'Ctrl-D' আছে। আবার একবার '$lekho' কমান্ড দিয়ে দেখুন, ফের নিখুঁত চিনে যাচ্ছে কমান্ডটা। কমান্ড কমপ্লিশন ব্যবহার করুন। কমান্ড প্রম্পটে '$le' টাইপ করে ট্যাব মারুন, হয় দেখবেন গোটা '$lekho'-টাই ও দেখিয়ে দিচ্ছে, নয়তো নিচে একাধিক সম্ভাবনা দেখাচ্ছে, যার একটা '$lekho'। মানে, এই ব্যাশ আমাদের বানানো লোকাল ভ্যারিয়েবলটাকে চেনে। এ তো চিনবেই, ভ্যারিয়েবলটা বানিয়েছিলাম এই ব্যাশ প্রসেসেই। অন্য ব্যাশ প্রসেসে পাওয়ার জন্যে এবার একে এক্সপোর্ট করুন। 'export lekho', কমান্ড দিন, এন্টার মারুন। ব্যাস। আপনি রপ্তানি করে দিলেন। এবার আগের মতই 'bash' টাইপ করে আর একটা ব্যাশ খুলুন, সেখানে দেখুন, এই '$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 হাত ঘুরে আসার কোনো ব্যবস্থা এখনো করেননি। এদের পকেট দেখবেন না, 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 ইউজারের হোমের লোকাল কনফিগারেশন ফাইল। যা দিয়ে 'lekho' ভ্যারিয়েবলটা বানিয়েছিলাম, সেই গোটা লাইনটা, কোট চিহ্ন সহ, নিট যোগ করে দিন আপনার হোমের '~/.bashrc' ফাইলে। এবার ঠিক অ্যালিয়াসের মতই, লগ-ইন থেকে শুরু করে কমান্ড প্রম্পটে ইন্টারাক্টিভলি চালু করা যে কোনো ব্যাশেই পাবেন লোকাল ভ্যারিয়েবল 'lekho'। তার মান আকারে '$lekho' কমান্ডটাও ব্যবহার করতে পারবেন। এর আগে অব্দি লোকাল ভ্যারিয়েবল 'lekho' ছিল অস্থায়ী একটি ব্যাশ-প্রসেসের নিজস্ব। সেই ব্যাশেই তার জন্ম ও মৃত্যু। এবার একে স্থায়ী করে দেওয়া হল, '~/.bashrc' ফাইলে তার জন্মপ্রক্রিয়াটাকে ভরে দিয়ে। আপনি যদি সিস্টেমের সুপারইউজার হন, চাইলে একে সিস্টেম জুড়ে যে ইউজারের কাছে প্রাপ্তব্য করে তুলতে পারেন। তখন আর একজন ইউজারের হোমে '~/.bashrc' ফাইলে নয়, লাইনটা যোগ করবেন ব্যাশের গ্লোবাল কনফিগারেশন ফাইলে। '/etc'-র গ্লোবাল কনফিগারেশন ফাইলে যোগ করার পর 'lekho' হয়ে যাবে গ্লোবাল ভ্যারিয়েবল, সিস্টেম জুড়ে যে কোনো ইউজারই তাকে পাবে। স্থায়ী ভাবে। যে কোনো ইউজারের যে কোনো ব্যাশ প্রসেস চালু হওয়ার আগে '/etc' ডিরেক্টরির কনফিগারেশন ফাইল পড়ে ব্যাশ তাকে বানিয়ে নেবে। মোট এনভায়রনমেন্টের একটা অংশ হয়ে যাবে, তাই 'lekho'-কে ডাকব এনভায়রনমেন্ট ভ্যারিয়েবল বলে।

ব্যাশের এই দুই ভ্যারিয়েবল, এনভায়রনমেন্ট এবং লোকাল, ব্যাশের কাছে আসে দুই ভাবে। এনভায়রনমেন্ট ভ্যারিয়েবল বানিয়ে তোলে সিস্টেম, '/etc/profiles' বা '/etc/bashrc' ফাইল অনুযায়ী। যেমন 'SHELL', 'PS1', 'PATH' ইত্যাদি। পরে বিশদে আসছি এদের কথায়। লোকাল ভ্যারিয়েবলদের বানিয়ে তোলে ইউজার, ইন্টারাক্টিভ ব্যাশ প্রক্রিয়ায়, প্রম্পটে কমান্ড দিয়ে, যেমন 'lekho'। বা, ইউজারের হোমের লোকাল কনফিগারেশন ফাইল পড়ে ইউজারের ব্যাশ-প্রসেস এদের বানিয়ে তোলে। লোকাল ভ্যারিয়েবলদের হদিশ থাকে ইউজারের '~/.profile' বা '~/.bashrc' ফাইলে। ব্যাশ চালু হওয়ার সময় ইউজারের লোকাল কনফিগারেশন অনুযায়ী এদের বানায়, যে কনফিগারেশনগুলো নিজের মর্জি বা প্রয়োজন মত ইউজার বদলে নিতে পারে। ঠিক যেমন 'lekho'-কে স্থায়ী করা হল কনফিগারেশন ফাইলে লিখে দিয়ে।

৬।। ব্যাশ ভ্যারিয়েবলের ঠিকুজি

কমান্ড প্রম্পটে যে কমান্ড দিয়ে আমরা 'lekho' ভ্যারিয়েবলটা বানালাম, একে বলে ভ্যারিয়েবলের ডেফিনিশন বা সংজ্ঞা। ভ্যারিয়েবল-ডেফিনিশনের তিনটে অংশ। প্রথম তার নাম, '*variable_name*'। তার পরে অ্যাসাইনমেন্ট অপারেটর বা ধার্যীকরণ মানে আমাদের চেনা 'সমান চিহ্ন', '='। তারপর ভ্যারিয়েবলের মান, '*variable_value*'। এখানে আমরা এই '=' চিহ্নকে অ্যাসাইনমেন্ট অপারেটর বলে ডাকছি। এটা খেয়াল করুন। পরে, শুধু শেলস্ক্রিপ্টে

নয়, যে কোনো কম্পিউটার ভাষাকে বোঝার জন্যেই আপনার এটা কাজে লাগবে। যখনই ভ্যারিয়েবলের নামটা দিলাম, সেই নামে স্মৃতিভূমিকে একখণ্ড জমি চিহ্নিত হল। অ্যাসাইনমেন্ট অপারেটর ব্যাশকে জানাল, ডানদিকে যে মান দেওয়া হচ্ছে সেটাকে অ্যাসাইন বা ধার্য করে দিতে হবে, ভরে দিতে হবে ওই ভ্যারিয়েবলের জমিটুকুতে। শুধু মাথায় রাখবেন, '=' চিহ্নের আগে এবং পরে কোনো স্পেস দেবেন না। স্পেস দিলে একে অ্যাসাইনমেন্ট অপারেটর বলে চিনতে পারবে না ব্যাশ। সংজ্ঞার ছকটা হল, '*variable_name=variable_value*'। এর সঙ্গে মেলান 'lekho'-র সংজ্ঞাটাকে। '*variable_name*' জমি, '*variable_value*' তার ফলন।

এই ব্যাশের বাইরে অন্য ব্যাশ প্রক্রিয়ায় বা অন্য প্রোগ্রামের কাছে চলটাকে এক্সপোর্ট করার কৌশল দেখলাম আমরা। '`export` *variable_name*' হল রপ্তানি আদেশের কাঠামো। এখানে যে ব্যাশ প্রসেস বা প্রোগ্রামের কথা বলছি, যাদের কাছে চলটা হাজির হবে, তারা সবাই এই ইউজারের, কারণ এটা আঞ্চলিক চল বা লোকাল ভ্যারিয়েবল। গ্লোবালি মানে গোটা সিস্টেম বা প্রতিটি ইউজারের কাছে একটা ভ্যারিয়েবলকে কী করে রুট হাজির করতে পারে, সেটাও জেনেছি আমরা।

সংজ্ঞা দিয়ে ভ্যারিয়েবল বানানো এবং তারপর তাকে অন্য প্রোগ্রামের কাছে রপ্তানি করা হয়ে গেল মানেই এই ইউজারের চালানো যে কোনো প্রোগ্রাম চলটাকে ব্যবহার করতে পারবে। ভ্যারিয়েবলের নামটা দিয়ে উল্লেখ করতে হবে, এবং নামের আগে লাগাতে হবে ওই '$' চিহ্নটা, তাহলেই প্রোগ্রামটা ভ্যারিয়েবলের মানটাকে পেয়ে যাবে। কোনো ভ্যারিয়েবলের মান জানার উপায় '`echo`'। আদেশ দিন, '`echo $`*variable_name*'। দেখুন স্ক্রিনে তার মানটা, '*variable_value*', ফুটে উঠেছে। '`echo $lekho`' আদেশ দিয়ে যেমন পেয়েছিলাম গোটা লাইনটা।

ভ্যারিয়েবলগুলোর কনফিগারেশন বদলানোর প্রথমতম কায়দাটাও জানি আমরা। কনফিগারেশন ফাইলে কোনো একটা লাইনকে অকেজো করে দিতে চাইলে লাইনটার আগে একটা '#' চিহ্ন লাগিয়ে দাও। পরে, ফেরত পেতে হলে, স্রেফ '#' চিহ্নটাকে উড়িয়ে দাও। একে বলে কমেন্টিং আনকমেন্টিং। মন্তব্যীকরণ অমন্তব্যীকরণ। 'মন্তব্য' বলতে যা মূল ফাইল নয়, ফাইলেরই কোনো বিষয় নিয়ে কোনো আলোচনা। '#' চিহ্ন থাকলে ব্যাশ ধরে নেবে, লাইনটা একটা মন্তব্য, আর পড়বেনা। এটা নানা জায়গাতেই বেশ কাজে আসে। একটা লাইনের কী কাজ সেটা বুঝতে লাইনটার আগে '#' চিহ্ন লাগিয়ে দিলাম। এবার দেখলাম কী হয়। একটা কনফিগারেশন ফাইলে বা একটা শেলস্ক্রিপ্টে কী আছে তার হদিশ পাই আমরা মন্তব্য থেকে। '/etc' ডিরেক্টরির কনফিগারেশন ফাইল বেশ কয়েকটা বুঝেছিলাম আমরা '#' চিহ্ন লাগানো কমেন্ট লাইন পড়ে। যে স্ক্রিপ্টগুলো আপনি নিজে বানাবেন, সেগুলোতেও এটা লাগাবেন। শুধু অন্যের না, নিজের জন্যেও। এটা অবশ্যম্ভাবী যে প্রথম শেলস্ক্রিপ্ট শেখার ধুমকিতে আপনি অজস্র কাজের অকাজের স্ক্রিপ্ট লিখবেন, যার বেশির ভাগ কলকব্জাই আপনি মেশিন অফ করার পর বেমালুম ভুলে যাবেন। পরে মাথার চুল ছিড়তে হবে এটা বুঝতে যে স্ক্রিপ্টটার কাজ কী, বেড়াল তাড়ায় না ঝাড়ে বাঁশ কাটে। প্রথম থেকেই শেলস্ক্রিপ্টে কমেন্ট লাগানো প্র্যাকটিস করুন। ভ্যারিয়েবলগুলোর নাম দেওয়ার সময়েও একটা জিনিষ খেয়াল রাখবেন। কম টাইপ করার স্বার্থে খুব ছোট ছোট নাম দিতেই পারেন, পরে কিছুতেই মনে করতে পারবেন না, কোন ভ্যারিয়েবলটা কী। নাম থেকেই যদি আন্দাজ করা যায়, এই ঝামেলাটা হয়না। নয় একটু বড় নামই হল। টাইপ করায় ভয় পেলে চলবেনা, মোটামুটি পরিমাণ কম্পিউটার ব্যবহার করার প্ল্যান থাকলেও, টাচ টাইপিং শিখে নিন, দশ আঙুলে টাইপ করা। প্রথম বছরেই মোট কাজের সময়ের এতটা সাশ্রয় হবে যে প্রাথমিক পরিশ্রমটা উশুল হয়ে যাবে। গ্নু-লিনাক্সে টাইপ শেখার প্যাকেজ জিটাইপিস্ট (gtypist) আছে, আগেই বলেছি, ভীষণ কাজের।

৭।। ব্যাশ ভ্যারিয়েবলের কনফিগারেশন

ব্যাশ ভ্যারিয়েবল কনফিগার করতে নেমে প্রথম জবাই করা যাক পুরোনো পাপী পথনির্দেশ বা 'PATH'। বারবার নানা কাজে নানা প্রসঙ্গে তাকে টেনে এনেছি, বদলে যেতে দেখেছি, কিন্তু নিজেরা কখনো বদলাইনি। বদলানোর দরকার প্রায়ই হয়, কখনো সাময়িক, কখনো স্থায়ী রকমে। ধরুন, একটা ফাইল বা প্রোগ্রাম আপনি আপনার কাজে লাগাতে চাইছেন, যা স্বাভাবিকভাবে আপনার পথনির্দেশে নেই। একটা উদাহরণ নেওয়া যাক। তার আগে, 'dd' নামের ইউজার হিশেবে ব্যাশের পথনির্দেশটা একবার দেখে নিই। '`echo $PATH`' কমান্ড দিয়ে পেয়েছিলাম, '/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 আলাদা আলাদা এগারোটা পথ আছে ':' চিহ্ন দিয়ে আলাদা করে করে। এই এগারোটা পথাংশ মিলে 'dd' নামক ইউজারের গোটা পথনির্দেশ। প্রথম উপাদানটা দেখুন, 'dd' নামক ইউজারের নিজের হোমের বাইনারি ডিরেক্টরি, '/home/dd/bin'। এই ডিরেক্টরিটা কিন্তু এমনিতে থাকেনা হোম ডিরেক্টরিতে। ইনস্টল হওয়ার সময় সুজে বা স্ল্যাকওয়ার কেউই বানায় না। কিন্তু পথনির্দেশে থাকে। ইউজার চাইলেই বানিয়ে নিতে পারে। নিজের বানানো কোনো প্রোগ্রাম বা শেলস্ক্রিপ্ট এখানে রাখলে শেল তাকে পেয়ে যাবে। প্রোগ্রামটা কোথায় আছে তার পুরো ঠিকানা আলাদা করে দিতে হবেনা। যেমন নয় নম্বর দিনে বারবার 'writefiles' ইত্যাদি যে শেল-স্ক্রিপ্টগুলো আমরা ব্যবহার করেছি, তাদের সবগুলোই রাখা আছে 'dd'-র নিজের বানানো '/home/dd/bin' ডিরেক্টরিতে। যে কোনো ডিরেক্টরি থেকেই কমান্ড প্রম্পটে জাস্ট `writefiles` কমান্ড দিলেই স্ক্রিপ্টটা চালাতে পারে ব্যাশ। শুধু '/home/dd/bin' না, এই এগারোটা পথাংশের যে কোনো একটায় থাকলেই সেটা হবে। কমান্ড প্রম্পটে যে কমান্ডই দিই-না কেন, হয় কমান্ডের প্রোগ্রামটার পুরো ঠিকানা দিয়ে দিতে হবে, নয়তো তাকে এই এগারোটা পথাংশের কোনো একটায় থাকতেই হবে। নয়তো ব্যাশ তাকে চালাতে পারবে না, হার্ডডিস্কের কোনো না কোনো ডিরেক্টরিতে প্রোগ্রামটা থাকা সত্ত্বেও। তখন কী হবে?

গুইতে বা গ্রাফিকাল ইউজার ইন্টারফেসে যখন জজজ-এ, মানে ইন্টারনেটে ব্রাউজ করি, যে ব্রাউজারটা সবচেয়ে বেশি ব্যবহার করি সেটা হল মোজিলা-এক্সএফটি (mozilla-xft)। এই প্যাকেজটা এমনিতে থাকেনা সিস্টেমে, নেট থেকে নামিয়েছিলাম। টার বিজিপটু থেকে বড় করে নিয়েছিলাম '/opt' ডিরেক্টরির মধ্যে, `tar xvjf` কমান্ড দিয়ে। এতে '/opt/mozilla-xft' নামে একটা ডিরেক্টরি তৈরি হয়ে গেছিল। মোজিলা-এক্সএফটি আমায় কম্পাইল করতে হয়নি, এটা একটা প্রিকম্পাইলড বাইনারি। মানে এর ভিতরে 'mozilla' নামে একটা বাইনারি বা প্রোগ্রাম ফাইল আগে থেকেই রাখা আছে, যেটা সুজে ৮.২ সিস্টেমে সবচেয়ে ভালো চলবে, টেকনিকাল ভাষায় অপটিমাইজড। এবার সমস্যাটা হল এই যে এই বাইনারিটার ঠিকানাটা দেখুন, '/opt/mozilla-xft/mozilla'। এই পথটা কিন্তু 'dd'-র '$PATH'-এ নেই। তাহলে 'dd' কী করবে? 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/mozilla-xft/mozilla'। নইলে ব্যাশ প্রতিবারই তার অজ্ঞতা জ্ঞাপন করবে, কমান্ড নট ফাউন্ড। প্রতিবারই এরকম দিতে হবে যতবার 'mozilla' চালাব। এর একটা সহজ উপায় হল, সাময়িক ভাবে 'PATH' ভ্যারিয়েবলটাকে লোকালি কনফিগার করে নেওয়া। এর জন্যে কমান্ড লাগবে দুটো, পরপর সেমিকোলন (`;`) দিয়ে দিতে পারি, ব্যাশ '`;`' চিহ্নকে যতি হিশেবে বোঝে, পরে আমরা আরো ব্যবহার করব।

```
PATH=$PATH:/opt/mozilla-xft ; export PATH
```

কমান্ড দুটোর দ্বিতীয়টাকে আমরা চিনি, অন্য ব্যাশ তথা প্রোগ্রামের কাছে কোনো ব্যাশ ভ্যারিয়েবলকে রপ্তানি করে। প্রথম কমান্ডটাও চিনি, সাত নম্বর দিনের ১.৩ নম্বর সেকশনে রুট হয়ে এমপ্লেয়ারে গান বাজাব কী করে তার কায়দা হিশেবে বলে এসেছিলাম, তখন ব্যাখ্যা করিনি, আপনাদের একটু আন্দাজ দেওয়ার চেষ্টা করছিলাম শেপ অফ থিংস টু কাম। এখন দেখুন, বুঝতে পারবেন, আসলে চালু 'PATH' ভ্যারিয়েবলটাকে আমাদের প্রয়োজন মোতাবেক বদলে দিচ্ছে দ্বিতীয় কমান্ডটা। আমরা জানি '=' একটা অ্যাসাইনমেন্ট অপারেটর, একটা বিশেষ মানকে একটা ভ্যারিয়েবলে অ্যাসাইন করে দেয়। '=' চিহ্নের বাঁদিকে থাকে ভ্যারিয়েবলের নাম, ডানদিকে কাঙ্খিত মান। ভ্যারিয়েবলের সংজ্ঞার ছক থেকে দেখে নিন। দ্বিতীয় কমান্ডে '=' চিহ্নের বাঁদিকে আছে 'PATH', ডানদিকে তার যে ভ্যালুটা করে নিতে চাইছি, মানে নতুন পরিবর্তিত '$PATH'। ডানদিকে দেখুন, দুটো অংশ। একটা যতিচিহ্ন ':', যা, আমরা দেখেছি, '$PATH' বা 'PATH' ভ্যারিয়েবলের মানে পরপর একাধিক অংশকে একটার সঙ্গে আর একটা যোগ করে যায়। এখানেও ':' চিহ্নটা 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, তারপর সেই বাড়তি পথাংশ যা যোগ করতে চাইছি, '/opt/mozilla-xft'। তার মানে দেখুন, এই কমান্ডটা যাত্রা শুরু করছে একটা '$PATH' নিয়ে, শেষ হচ্ছে আর একটা '$PATH'-এ। এই নতুন '$PATH' হল পুরোনো '$PATH' যোগ ওই বাড়তি পথাংশ '/opt/mozilla-xft'। কিন্তু, এই কমান্ডের পর নতুন '$PATH' যে সত্যিই বদলে গেছে তার প্রমাণ পাব কী করে? সহজ। আবার ব্যাশকে প্রতিধ্বনি করতে বলা, `echo $PATH`।

এবার যে মানটা পাব আমরা তাতে ওই বাড়তি পথাংশ যোগ হয়ে গেছে। নতুন '$PATH' দাঁড়াল '/home/dd/bin:/usr/local/bin:/usr/bin:/usr/X11R6/bin:/bin:/usr/games:/opt/gnome2/bin:/opt/gnome/bin :/opt/kde3/bin:/usr/lib/java/jre/bin:/opt/gnome/bin:/opt/mozilla-xft'। দেখুন আগের '$PATH' পুরো আছে, তার সঙ্গে আমাদের যোগ করা বাড়তি অংশটা এসে গেছে একদম শেষে। মোট পথাংশ এগারোটার জায়গায় বারোটা হয়ে গেছে। তার বারোতমটা আমাদের যোগ করা '/opt/mozilla-xft'।

এতে লাভটা কী হয়েছে? এখন থেকে যখনি কমান্ড প্রম্পটে '`mozilla`' কমান্ডটা ব্যবহার করব, ব্যাশ আর বলবে না, কমান্ড নট ফাউন্ড। পরপর খুঁজতে খুঁজতে বারো নম্বর অংশটা খোঁজার সময়েই ব্যাশ পেয়ে যাবে তার আরব্ধ কমান্ড, '/opt/mozilla-xft' ডিরেক্টরি থেকে 'mozilla' কমান্ডটা। কারণ, আমাদের চাওয়া ওই পথাংশ ব্যাশের '$PATH'-এ এসে গেছে। কিন্তু, মাথায় রাখুন, এই বদলটা সাময়িক, অস্থায়ী। একবার লগ-আউট করে গেলে পরের বার এটা আর পাওয়া যাবেনা। তখন ব্যাশ আবার '$PATH' বলতে প্রথম মানটাই বুঝবে। এবার, আপনি অফিসিয়াল রকমে এই বদলটা যদি পাকাপোক্ত করে নিতে চান, তার উপায় কী? সেটাও এখন চেনা আমাদের। ইউজারের হোম ডিরেক্টরির লোকাল কনফিগারেশনে ঢুকিয়ে নিতে পারে ইউজার, বা রুট এটাকে গ্লোবাল কনফিগারেশন ফাইলেও যোগ করে দিতে পারে। তাহলে সিস্টেম জুড়ে সকলেই মোজিলা-এক্সএফটি ব্যবহার করতে পারবে।

একসময়কার পুরোনো ডস আর উইনডোজের অভিজ্ঞতা যাদের আছে তারা 'autoexec.bat' নামে ব্যাচ ফাইলের কাজের সঙ্গে ব্যাশ কনফিগারেশন ফাইলের মিলটা নিশ্চয়ই পাচ্ছেন। ব্যাচ ফাইলকে একরকম দুধেভাতে শেলস্ক্রিপ্ট বলতে পারেন। সেখানেও মনে করে দেখুন, প্রথম দুটো বিষয়ই থাকত পথ আর প্রম্পট। তার পরে আসত কটা ফাইল একসঙ্গে খোলা যাবে তার হদিশ। সেসব বিটলেপনা গ্নু-লিনাক্সে নেই। এখানে মেমরি এবং ভারচুয়াল মেমরি অমন খাজা রকমে নিয়ন্ত্রণ করা হয়না। প্রথম উইন্ডোজ থেকে গ্নু-লিনাক্সে এসে চমকটা লাগে ওই কারণেই। একসঙ্গে এতকিছু এতগুলো কাজ এত চমৎকার করা যায়, এই মেশিনেই, কই উইন্ডোজে তো হতনা? মাল্টিটাস্কিং-টা ডস বা উইন্ডোজ সিস্টেমে আরোপিত, প্রক্ষিপ্ত, বাইরে থেকে চাপানো। 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 আসলে মাল্টিপ্লেক্সিং-এর মধ্যে দিয়েই তৈরি হয়েছে। এক সঙ্গে অনেক ইউজারের অনেক প্রসেসকে মেলাবেন তিনি মেলাবেন বলেই তৈরি হয়েছিল ইউনিক্স, এবং তারই উত্তরাধিকারে গ্নু-লিনাক্স। তাই ঝোড়ো হাওয়া আর পোড়ো বাড়িটার ওই ভাঙা দরজাটা সেখানে জোর করে এমন মেলাতে হয়না যাতে একটা গান বাজলেও ডিফ্র্যাগমেন্টেশন কোঁত পাড়তে থাকে, বারবার নতুন করে শুরু হয়, নিজেই গাইতে শুরু করে, আমারে তুমি অশেষ করেছো। অথচ গ্নু-লিনাক্সে একটা আপডেট-ডিবি এবং একটা হার্ডডিস্ক ব্যাকআপ ব্যাকগ্রাউন্ড প্রসেসে নিয়ে, ম্যাট্রিক্স সিনেমাটা চালিয়ে দেখেছি, পাশে তার চিত্রনাট্যটা খোলা, আর ইম্যাক্সে তার নোট নিচ্ছি, অন্য সিডি ড্রাইভ থেকে এমপ্লেয়ার অ্যাপোক্যালিপ্স নাউ এনকোড করে এভিআই বানাচ্ছে। আসলে দেখতে চাইছিলাম, কতদূর টানতে পারে সিস্টেম। কিয়ানু রিভস এমন একটা হেঁচকি অব্দি তুলল না যা চিত্রনাট্যে নেই, মাইরি, মিলিয়ে বলছি। চিত্রনাট্যটা নেট থেকে নামানো, অরিজিনাল।

পথনির্দেশ বা 'PATH'-এর মত আর একটা ব্যাশ ভ্যারিয়েবল হল খোদ কমান্ড প্রম্পটটাই, মানে তার তার আক্ষরিক আকারটা। পাঁচ নম্বর দিনে আমরা ম্যানড্রেকের কমান্ড প্রম্পটটা দেখিয়েছিলাম, মনে পড়ছে? ডিস্ট্রো থেকে ডিস্ট্রোয় কমান্ড প্রম্পটের ডিফল্ট আকারটা একটু-আধটু আলাদা হয়। এছাড়া, চাইলে সেটা নিজের মত করে বদলে নেওয়া যায়। যে ভ্যারিয়েবলটা ঠিক করে দেয় কমান্ড প্রম্পটের আকার তার নাম 'PS1'। 'PATH'-এর মতই আর একটা এনভায়রনমেন্ট ভ্যারিয়েবল, যাকে আপনি লোকালি বদলাতে পারেন। অন্য ইউজারের প্রম্পট তাতে বদলাবে না। সুজে সিস্টেমে কমান্ড প্রম্পটটা দেখুন, এটা ইউজার 'dd'-র। অন্য ইউজারে ওই '`dd`' অংশটা বদলে যাবে।

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

এবার দেখা যাক 'PS1' ভ্যারিয়েবলের কী মান কমান্ড প্রম্পটটাকে এই আকার দিচ্ছে। '`echo $PS1`' কমান্ড দেওয়ায় ব্যাশ স্ক্রিনে ফুটিয়ে তুলল '`\u@\h:\w>`'। হঠাৎ করে দেখলে মনে হতেই পারে, ব্যাশ আমায় হিব্রু ভাষায় গালাগাল দিচ্ছে। কিন্তু তা নয়, এই '$PS1'-এর শেষ '>' চিহ্নটা কমান্ড প্রম্পটের সঙ্গে মিলিয়ে নিন।

দুটোতেই এক। আরো একটা চিহ্ন একদম এক। সেটা '@', অ্যাকাউন্টান্সিতে যা ছিল 'অ্যাট-দি-রেট-অফ', এখন নেট পরিভাষায় 'অ্যাট'। এখানে ঠিক 'অ্যাট' এই অর্থেই ব্যবহার হচ্ছে। নেটের লিংকগুলো, ওয়েবসাইটের মধ্যেকার বিভিন্ন ঠিকানা এমনকি উইন্ডোজেও দেখায়, খেয়াল করবেন, '/' যতিচিহ্ন ব্যবহার করে। উইন্ডোজের পথনির্দেশের স্বাভাবিক যতিচিহ্ন কিন্তু '\'। মনে করুন, ছয় নম্বর দিনে তিনরকম ওএস-এর তিনরকম যতিচিহ্নর কথা ছিল। আসলে ইন্টারনেট, ওয়ার্ল্ড-ওয়াইড-ওয়েব এই গোটাটাই গজিয়ে উঠেছে ইউনিক্স জগত থেকে। সেই ব্যাকরণ মানতেই হয়। নিজের সিস্টেমে যে পথনির্দেশই দেখাক, ওয়েবসাইটের বেলায় উইন্ডোজকেও '/' দিয়েই পথ দেখাতে হয়। এই বিষয়টা আরো স্পষ্ট হবে '$PS1'-এর '@' চিহ্নের ঠিক আগের আর পরের '\u' এবং '\h' অংশদুটোর মানে জানলে। ম্যানুয়াল দেখে এখনি জেনে যেতে পারেন। না-জেনেই বোঝার চেষ্টা করা যায় কিনা দেখুন তো। তৃতীয় বিশ্ব একটা রকমে, আমরা তো সবসময়ে জরুরি ম্যানুয়ালটুকুও পাইনা। দেখুন, প্রম্পট মিলিয়েই বোঝা যাচ্ছে, '@' চিহ্নের বাঁদিকের অংশটা ইউজারের নাম। অর্থাৎ, '\u' নিশ্চয়ই ইউজারের নাম। বাকি রইল '\h', প্রম্পটে মিলিয়ে দেখা যাচ্ছে যার আকার 'mahammad'। মনে করুন, এই নামটা পেয়েছিলেন হোস্টনেম কনফিগারেশনের ফাইলের সূত্রে, নয় নম্বর দিনে। হোস্টনেমের 'h' আর এই 'h' মিলে যাচ্ছে, তাহলে এটা নিশ্চয়ই হোস্টনেম। এরপরে একটা '~' চিহ্ন। এটা আমরা চিনি, ইউজারের হোম-ডিরেক্টরির ব্যাশ-সঙ্কেত। এবার ব্যাশের ম্যানুয়াল মেলান, ঠিক ভেবেছি কিনা। একটা সেকশন '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
```

'echo $PS1' করে আমরা যা পেয়েছিলাম, '\u@\h:\w>', তার গোটাটাই মেলানো যাচ্ছে। এখন আমরা আরো কয়েকটা ফরমুলা জানি, 'PS1' ভ্যারিয়েবলের। এবার, সেই ফরমুলা মাফিক প্রম্পট বদলে দেখা যাক। কমান্ড দেওয়া যাক, 'PS1="\u@\h<\w>\t>"', যাতে আগের প্যারার ফরমুলা মোতাবেক আগের প্রম্পটের উপাদানগুলোর সঙ্গে এখন সময়টাও দেখায়। ঠিক তাই হল। প্রম্পটের আকার বদলে গেল।

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

চালু কাজের ডিরেক্টরিটাকে আমরা দুপাশে দুটো '<' আর '>' চিহ্নের মধ্যে দিতে বলেছিলাম, ঠিক তাই করেছে ব্যাশ। আর তারপর দিতে বলেছিলাম '\t', মানে সময়, চব্বিশ ঘন্টার ফরম্যাটে, মানে দুপুর একটা যেখানে তেরোটা, ট্রেনের সময়ের মত। এরকম যা মন চায় দিতে পারেন। অজস্র অপশন আছে ব্যাশের ম্যানুয়ালে। প্রম্পটের এই বদল অস্থায়ী বদল। একে স্থায়ী করবেন কী করে? ঠিক 'PATH' ভ্যারিয়েবলের মত, 'PS1' ভ্যারিয়েবলও ইউজারের ব্যাশে স্থায়ী হবে তখনই যখন একে লোকাল কনফিগারেশনে তুলে দেবেন। গ্লোবাল এনভায়রনমেন্ট ভ্যারিয়েবল তখনই হয়ে উঠবে যখন এর ঠাঁই হবে গ্লোবাল কনফিগারেশন ফাইলে। যা করতে পারে শুধু রুট।

৮।। এক কুচো ব্যাশ ফাংশন

আজই বলেছি, ব্যাশ 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, যাদের প্রায়ই একসঙ্গে করতে হয়। সমাহারটা যাতে যে কোনো প্রোগ্রাম কাজে লাগাতে পারে তার দরকার মত, তাই কিছু নির্দিষ্ট ফাংশনকে ব্যাশের কনফিগারেশন ফাইলেই দেওয়া থাকে। ফাংশন এবং লাইব্রেরি নিয়ে আলোচনা মনে আছে? ব্যাশ কনফিগারেশন ফাইলকে সেই অর্থে ব্যাশ ফাংশনের একটা আনবিক লাইব্রেরি ভাবা যায়। গ্নু-লিনাক্সে ব্যাশ সর্বব্যাপী হওয়ায় শুধু ব্যাশ না, অন্য যে কোনো প্রোগ্রাম বা যে কোনো ইউজার নিজের কাজে ফাংশনগুলো ডেকে নিতে পারে। ধরুন, আপনার লেখা প্রবন্ধগুলোকে আপনি প্রায়ই 'bzip2' করে '.bz2' এক্সটেনশনের ফাইল করেন। করতেই হয়, কারণ, পাঠক পাওয়ার আগ্রাসী আকাঙ্খায় আপনি তাদের ইমেল করেন, এবং '.bz2' মানে, টেক্সট ফাইল হলে, মোটামুটি মূল ফাইলের দশ থেকে কুড়ি শতাংশ। মানে, এই 'bzip2' অনলাইন খরচ অনেকটা বাঁচিয়ে দিচ্ছে আপনার। তাই রেগুলার আপনি ব্যবহার করেন। একটা ফাংশন আপনি বানিয়ে রাখতে পারেন

আপনার '~/.bashrc' ফাইলে, যাতে বারবার গোটা 'bzip2' কমান্ডটা না-দিয়ে গোটা কাজটাই এককথায় হয়ে যায়। এই ফাংশনটা আপনি যোগ করে নিতে পারেন আপনার '~/.bashrc' ফাইলে। যোগ করার পর একবার 'Ctrl-D' বা 'exit' করে বেরিয়ে এসে ফের লগ-ইন করে নিলেই ফাংশনটা পাবেন আপনার যে কোনো কাজে। লগ-আউট/লগ-ইন করার পাবেন না, কারণ নতুন '~/.bashrc' ব্যাশ তখনো পড়েনি। ব্যাশ ম্যানুয়ালে একটা অন্য উপায় আছে লগ-আউট না করেই কনফিগারেশন ফাইল ব্যাশকে দিয়ে পড়িয়ে নেওয়ার, 'source'। ভালো মনে পড়ছে না, ম্যান দেখে নিন। এবার এই 'bzip2'-র কাজের জন্যে খুব সরল একটা ফাংশন বানানো যাক।

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

অর্থাৎ, 'packall()' ফাংশন বানিয়ে ফেললাম আমরা। '~/.bashrc'-তে ফাংশনটা যোগ করে লগ-আউট/লগ-ইনের পর থেকে 'packall' কমান্ড দিলেই গোটা ডিরেক্টরি জুড়ে প্রতিটা ফাইল এক একটা আলাদা '.bz2' ফাইল করে দেবে ব্যাশ। এবং এটা দিয়ে আপনি আপনার ডিস্কভূমিও বাঁচাতে পারেন, গোটা ডিরেক্টরির সাইজও অনেক ছোট করে দেয়, কারণ প্রতিটি ফাইলই সাইজে অনেক ছোট হয়ে গেছে। আর 'bzip2' করেছে অপশন '-9' দিয়ে, মানে সর্বোচ্চ সম্ভব কুঁকড়ে দিয়েছে। আপনি যদি একসাথে সব ফাইল 'bzip2' না-করে এক একটা ফাইল আলাদা করে করতে চান, তাহলে আপনাকে অন্য একটা ফাংশন বানাতে হবে, ধরুন 'packone()' নামে। এই ফাংশনটাও একই ভাবে '~/.bashrc' ফাইলে যোগ করে দিতে হবে।

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

একটা জিনিষ খেয়াল করুন, প্রত্যেকটা ফাংশন হেডিং-এ, যেখানে নাম দিয়ে ফাংশনটা শুরু হচ্ছে, আমরা ফাংশনের নামের পরেই একটা জোড়া ব্রাকেট '()' দিচ্ছি। এটাই নিয়ম। এটা দিয়েই ব্যাশ একটা ফাংশনকে চেনে। ফাংশনটা ব্যবহার করার সময় কিন্তু ব্রাকেট দিতে হবেনা, শুধু ফাংশনের নাম দিলেই চলবে। 'packone()' ফাংশন এখন এসে গেল ব্যাশে। 'packall()' ফাংশনটার সঙ্গে তফাত এই যে এটা আর ডিরেক্টরি ধরে কাজ করেনা, ফাইল ধরে করে। এবং কমান্ড দেওয়ারও একটা তফাত ঘটেছে। আগেরটায় 'packall' কমান্ড দিলেই চলত, পরের 'packone()' ফাংশনটার জন্যে প্রথমে দিতে হবে 'packone', তার পরে একটা ফাইলের নাম, তবে কমান্ডটা সম্পূর্ণ হবে। তখন ব্যাশ সেই ফাইলটাকে কুঁকড়ে একটা '.bz2' ফাইল বানিয়ে দেবে। যা দিয়ে এই ফাইলের নামটা দেওয়ার ব্যাপারটা করা হচ্ছে, সেই জিনিষটা এখনো আমাদের অজানা, '$1'। একটু বাদেই আমরা দেখব, এর মানে, 'packone' কমান্ডটার পরে যে ফাইলটার নাম আপনি দিচ্ছেন, '$1' দিয়ে ব্যাশ তাকেই বুঝছে। এটা ব্যাশের একটা রেডিমেড ফরমুলা। ব্যাশ প্রম্পটে কোনো কমান্ড দেওয়া মানেই তার পরের শব্দটা হল '$1'। ধরুন, 'bzip2' করতে চাইছেন 'essay.4.text' ফাইলটা, কমান্ড দেবেন 'packone essay.4.text'। ব্যাশ '$1' বলে চিনে নেবে ফাইলটা, এবং 'packone' কাজে লাগিয়ে সঙ্গে সঙ্গে 'essay.4.text.bz2' নামে একটা ফাইল বানিয়ে দেবে। এখন আর গোটা ডিরেক্টরি না, শুধু একটা ফাইল। কোনো ফাইলনাম না-দিয়ে, শুধু 'packone' কমান্ড দিলে ব্যাশ জানাবে —

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

সিস্টেম বোঝার একটা ভালো ব্যায়াম গন্ডগোলের খবর বা এরর মেসেজগুলো পড়া 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। মুহূর্মুহূ ফাইল কোঁকড়ালে, তাদের তো বড় করতেও হবে। ধরুন তার নাম দিলেন 'unpack()'। এই ফাংশনের লাইনগুলোও যোগ করে দিতে হবে '~/.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'। এই গোটাটাই নিজের খুশি মত, ঠিক ফাংশনের নাম যেমন। এই 'unpack()' ফাংশনটা ঠিক 'packall()' ফাংশনের মতই। 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। সেই তালিকার প্রথম ফাইলনাম হয়ে দাঁড়াচ্ছে ব্যাশের 'pack' ভ্যারিয়েবলের প্রথম মান, দ্বিতীয় ফাইলনাম হচ্ছে দ্বিতীয় মান, এই ভাবে এগোচ্ছে। নয় নম্বর দিনে ব্যাশস্ক্রিপ্ট ধরে এটা বুঝেছি আমরা। 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। ভালো করে ব্যাশাধিকারী হতে চাইলে ফাংশন বুঝতেই হবে। এটা তার বিসমিল্লা।

৯।। ব্যাশ একটা স্ক্রিপ্টিং ল্যাংগুয়েজ

ব্যাশে কমান্ড প্রম্পটে যে সব কমান্ড দিয়ে আমরা নানা কাজ করছি, সেই কমান্ডগুলোকে একসঙ্গে গেঁথে দেওয়ার নানা কায়দা হয়। ফাংশনও একটা গ্রথিত কমান্ডসমাহার। কিন্তু তার সঙ্গে ব্যাশস্ক্রিপ্টের মূল পার্থক্য হল স্বাধীন স্বতন্ত্র অস্তিত্বের। ফাংশন একটা কমান্ডগুচ্ছ যা স্বনির্ভর এবং স্বতন্ত্র নয়। তাকে ডেকে আনে অন্য প্রোগ্রাম বা শেল। কমান্ড লাইন থেকে ডাকা যায় তাদের, যেমন আমরা ডাকলাম। শুধু কমান্ড প্রম্পট না, ফাংশনগুলোকে ডেকে নিতে পারে কোনো ব্যাশস্ক্রিপ্টও। যেই ডাকুক, তার চলার ভিতর থেকে গজিয়ে ওঠে ফাংশনটার চলা, ব্যাশের চলা, প্রোগ্রামের চলা, স্ক্রিপ্টের চলা। ফাংশনটা নিজে নিজে চলেনা। ফাংশনকে আলাদা করে চালাতে গেলে তাকে একটা আলাদা স্ক্রিপ্ট বানিয়ে নিতে হয়। সেই কায়দাটা আমরা আগেই শিখেছি। '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 ব্যাশস্ক্রিপ্ট হয়ে গেল। তার কাজ হল হুবহু ওই 'unpack()' ফাংশনটার সঙ্গে একই, কিন্তু সে এখন স্বতন্ত্র স্বাধীন একটা ব্যাশস্ক্রিপ্ট। শুধু, দেখুন, ব্যাশস্ক্রিপ্ট বানানোর সময় আমরা কিছু কমেন্ট যোগ করেছি, বলেছি তো, সেটাই প্রথা। আর ফাইলনামে '.sh' এক্সটেনশনটাও একটা প্রথা, চলে আসছে, একটা অভ্যস্ততা। তবে যে নামই দিন ব্যাশ স্ক্রিপ্ট হিশেবে কাজ করতে তার কোনো অসুবিধে হবেনা। রুট এই 'unpack.sh' ফাইলটা '/bin' ডিরেক্টরিতে রেখে দিলে ব্যাশ শেলের পথনির্দেশে চলে আসবে, কমান্ড প্রম্পটে ইউজার '`unpack.sh`' কমান্ডটা দিলেই ডিরেক্টরির যাবতীয় কোঁকড়ানো ফাইল বড় করে দেবে ব্যাশ।

```
#!/bin/bash
# Bashscript named 'unpack.sh'
# Expands all 'bz2 ' files in the directory
for pack in *.bz2
    do bunzip2 $pack
done
```

'unpack.sh'-কে ভাবুন। যত অকিঞ্চিতকরই হোক না, নিজের মত করে সে একটা কাজ করতে পারে, কোন ফাইলকে বাছব, ফাইলকে কী করব, এই নিয়ন্ত্রণটুকু নিজের হাতে না করে আমি মেশিনের হাতে ছেড়ে দিতে পারছি এই স্ক্রিপ্টের দৌলতে। এটা একটা প্রোগ্রাম। একে লিখলাম ব্যাশে। ব্যাশ শুধু একটা কমান্ড প্রম্পট না, একটা প্রোগ্রাম করার ভাষা, স্ক্রিপ্টিং ল্যাংগুয়েজ। একে 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 হল। তাকে কম্পাইল করলেন, কম্পাইলিং হল। 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 হল। এবং সবশেষে পোকা ছাড়ানো ফাইনাল মনোমত কোড থেকে কম্পাইল করে চূড়ান্ত চালনীয় ফাইল বা এক্সিকিউটেবল ফাইল বানালেন — গোটাটাকে মিলিয়ে হল প্রোগ্রামিং। এর সঙ্গে স্ক্রিপ্টিং-এর কিছু মৌলিক তফাত আছে। স্ক্রিপ্টিং বলতে আমরা বোঝাই, কিছু ব্যাশ বা অন্য কোনো শেলের কমান্ডকে নিচ্ছি, তাদের একত্রে সমাহৃত করে একটা ফাইলে লিখে ফেলছি স্ক্রিপ্ট লেখার নিয়মকানুন মেনে, যেমন গোড়াতেই '`#!/bin/bash`' লাইনটা লাগিয়ে নিচ্ছি ইত্যাদি, ফাইলটা লেখা শেষ হলে তাকে '`chmod`' দিয়ে এক্সিকিউটেবল করে নিচ্ছি, তারপর তাকে চালানোর চেষ্টা করছি, যথারীতি পোকা ছাড়াচ্ছি, আর সব শেষে চূড়ান্ত ফাইলটা চালাচ্ছি — এই গোটাটাকে মিলিয়ে বলে স্ক্রিপ্টিং।

পাঠমালার এক নম্বর দিনে আমাদের কম্পিউটারকে, মেশিনটাকে, 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। কম্পাইলার যা করে। আমাদের প্রখর বুদ্ধি আর মেশিনের জটিল এবং মহার্ঘ নির্বুদ্ধিতার মধ্যে দোভাষির কাজ করে কম্পাইলার। কিন্তু স্ক্রিপ্টের তো কম্পাইলার হয়না। তাহলে কাজ করছে কী করে? চেনা পুরোনো গামা মেশিনটাকে এমন বৈপ্লবিক ব্রেনোলিয়া খাওয়াল কে? 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 ভাষার প্রাথমিক পার্থক্যটা এই যে প্রোগ্রামিং ভাষা স্বভাবত অনেক বেশি শক্তিশালী। কাজও করে অনেক দ্রুত। প্রোগ্রামিং ভাষার উদাহরণ সি, ফোর্ট্রান, সি++, জাভা ইত্যাদি। এগুলোয় লেখা সোর্স-কোড কম্পাইল করে বাইনারি বানানো এবং ওএস থেকে ওএস-এ সেই বাইনারির স্থানান্তরযোগ্যতা বা পোর্টেবিলিটি নিয়ে আমরা আলোচনা করেছি চার পাঁচ ছয় এবং আট নম্বর দিনে। একই সোর্স-কোড থেকে একটা ওএস-এর জন্যে বানানো এক্সিকিউটেবল অন্য ওএস-এ চলেনা। আবার নতুন করে ওই ওএস-এর মত করে তাকে কম্পাইল করে নিতে হয়।

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-কোড একত্রে প্রসেস করে। ইন্টারপ্রিটার এগোয় আদেশ বাই আদেশ। একটা আদেশ ধরো পড়ো করো, তারপর আর একটা। কিন্তু এই রকম ভেঙে ভেঙে পড়ার কারণে তার গতিটাও কম্পাইলড ভাষাগুলোর তুলনায় অনেক কমে যায়। আবার এর সুবিধে এই যে, কমান্ডগুলোকে বুঝতে পারে এমন কোনো শেল ওএস-এ থাকলেই স্ক্রিপ্টটা চালানো যাবে, আলাদা কোনো সফটওয়ারের দরকারই পড়বে না। 'mplayer' তথা অন্য প্যাকেজের সোর্স-কোড থেকে বাইনারি কম্পাইল করার কথা এসেছে। যার প্রথম স্টেপে সোর্স-কোডের ডিরেক্টরিতে দাঁড়িয়ে '`./configure`' কমান্ড দিতে হয়, ওখানেই থাকে 'configure' স্ক্রিপ্টটা, আর কারেন্ট ডিরেক্টরি থেকে কিছু চালাতে গেলে গ্নু-লিনাক্সে '`.`' তো দিতে হবেই। এই 'configure' একটা স্ক্রিপ্ট যা গোটা মেশিনের হার্ডওয়ার এবং সফটওয়ার কনফিগারেশনের খুঁটিনাটি বুঝে সেই অনুযায়ী জিসিসি কম্পাইলার দিয়ে কম্পাইল করার বন্দোবস্ত করে। দ্বিতীয় এবং তৃতীয় স্টেপে আসে '`make`' এবং '`make install`'। এইমাত্র কনফিগার করে যে তথ্যগুলো পাওয়া গেল, তার সঙ্গে মিলিয়ে সোর্স-কোডকে জিসিসি দিয়ে কম্পাইল করে বাইনারি ফাইল তৈরি করে 'make'। এবং সেই বাইনারি ফাইল গ্নু-লিনাক্স ফাইলসিস্টেম-প্রথা মেনে ব্যাশের পাওয়ার মত পথের ডিরেক্টরিতে রেখে দেয় 'make install'। 'make' একটা ইউটিলিটি প্যাকেজ, গ্নু-র, রিচার্ড স্টলম্যান এবং রোলাঁ ম্যাকগ্রার বানানো। একটু মেক-এর খেজুরগাছে চড়ে আসতে পারেন, তাতে আপনারই উপকার হবে। খুব ভালো একটা বই আছে, 'লিনাক্স ডিভেলপমেন্ট প্ল্যাটফর্ম', রফিক উর

রহমান এবং ক্রিস্টফার পলের লেখা, গ্নু-লিনাক্স সিস্টেমে প্রোগ্রামিং নিয়ে, নেটে ফ্রিতে নামানো যায়। তাতে মেক নিয়ে বিরাট একটা চ্যাপ্টার আছে। মেক কাজ করে মেকফাইল (makefile) দিয়ে, যে মেকফাইল রাখা থাকে সোর্স কোডের ডিরেক্টরিতে। মেকফাইল একটা গাইড, কী ভাবে কম্পাইল করতে হবে। সেই ফাইল মেনে কম্পাইল করে বাইনারি বানায় জিসিসি। এবং 'mplayer' গোছের অনেক প্যাকেজেরই বাইনারি '/usr/local/bin' ডিরেক্টরিতে রেখে দেয় 'make install'। এই ডিরেক্টরি ব্যাশের পথের মধ্যে পড়ে, যে ডিরেক্টরিতে বসেই বাইনারিটা চালানোর কমান্ড দেওয়া হোক, তাকে ব্যাশ চালিয়ে দেয়। আমার মেশিনের সুজে ৮.২ সিস্টেমে 'mplayer' বাইনারি কম্পাইল করে রাখা ছিল। অর্থাৎ, বাইনারিটা কম্পাইলড হয়েছিল ওই সিস্টেমের যাবতীয় কনফিগারেশন মেনে। সুজে সিস্টেমের '/usr/local/bin' ডিরেক্টরি থেকে 'mplayer' বাইনারিটা আমার মেশিনেরই স্ল্যাকওয়ার সিস্টেমে কপি করে নিয়ে সেখানে চালাতে চাওয়ায়, চলল না। ব্যাশ প্রমাদ-বার্তা বা এরর মেসেজ দিল স্ক্রিনে।

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

অথচ, একই নিয়ম মেনে স্ল্যাকওয়ারে কম্পাইল করা 'mplayer' বাইনারি সেখানের '/usr/local/bin' ডিরেক্টরি থেকে দিব্য চলছে। এখানে আমি আপনাদের দুটো প্রশ্ন করব, যার একটারও উত্তর দেবনা। এখানে প্রমাদবার্তাটার ব্যাখ্যা আমি ইচ্ছে করেই দিলাম না, আপনি নিজে বোঝার চেষ্টা করুন। আর, যখন আমি সুজের '/usr/local/bin' থেকে স্ল্যাকের '/usr/local/bin'-এ 'mplayer' বাইনারিটা 'cp' করেছিলাম, তখন আমার হুবহু কমান্ডটা কী ছিল? আমি তো সামনে নেই, নিজেকেই দিন উত্তরগুলো। এবার, এই গোটাটায় আমরা কী পেলাম? প্রোগ্রামিং ভাষায় সি-তে লেখা 'mplayer' প্যাকেজের সোর্স কোড কম্পাইল করে একটা সিস্টেমে বানানো বাইনারি অন্য সিস্টেমে চলে-না। এমনটাই ঘটার কথা, ঘটানোর জন্যেই 'configure' করা। অন্য ওএস-এ চলবেনা, এবং নিজেরটায় খুব ভালো করে চলবে। কিন্তু, যে কোনো শেল স্ক্রিপ্ট যে কোনো ওএস-এ চলবে, যদি শেলটা থাকে। আর ব্যাশ তো থাকেই, যে কোনো গ্নু-লিনাক্স সিস্টেমে। সুজেতে বানানো স্ক্রিপ্ট স্ল্যাকে ঠিক স্ল্যাকে বানানো স্ক্রিপ্টের মতই চলবে। আবার, এই হার্ডওয়ার নির্ভরতা নেই বলেই হার্ডওয়ার নিপুণতাও নেই। খুব বড় কাজ শেল স্ক্রিপ্টে করতে গেলে মেশিনের রসদের উপর খুবই চাপ পড়ে। যা কম্পাইলড প্রোগ্রামে হয়না, কারণ, হার্ডওয়ারকে নিজের শিরায় নিয়েই সে তৈরি হয়েছে। তবে সচরাচর জাম্বো সাইজের ব্যাশস্ক্রিপ্টে করাও হয়না, পার্ল (Perl), লিস্প (Lisp), টিসিএল (Tcl) ইত্যাদি বরং ব্যবহার হয়। এক নম্বর দিনে মেশিনকে গাধা নামে ডাকার সেই কুচো সি প্রোগ্রামটা মনে আছে, এবার ওই কাজটাই করব ব্যাশ দিয়ে। তার আগে ওই সি প্রোগ্রামটা একবার দেখে নিন।

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

কোডটাকে 'gadha.c' নামে সেভ করেছিলাম আমরা। '.c' এক্সটেনশনটা সি-প্রোগ্রাম হিশেবে চিনে নেওয়ার চালু প্রথা। জিসিসি (gcc) দিয়ে এবার কম্পাইল করে নিতে হবে 'gadha.c' নামের এই সোর্স-কোডটাকে। জিসিসি কম্পাইলার গ্নু-লিনাক্স সিস্টেমে ইনস্টল করাই থাকে, ইনফ্যাক্ট আগেই তো বলেছি, গ্নু-লিনাক্স গজিয়েই উঠেছে গ্নু-র জিসিসির উপর দাঁড়িয়ে, লিনাসের কারনেল নিয়ে। 'gadha.c' যে ডিরেক্টরিতে আছে সেখানে দাঁড়িয়ে কমান্ড দিতে হবে, 'gcc -o gadha gadha.c'। জিসিসি তাতে এই ডিরেক্টরিতেই 'gadha' নামে একটা বাইনারি ফাইল বানিয়ে দেবে। এই '-o' অপশনটা দিয়ে বাইনারিটার জন্যে 'gadha' নামটা নিয়ে দেওয়া হল জিসিসিকে। না-দিলে জিসিসি বাইনারি ফাইলটাকে দেবে জিসিসির ডিফল্ট নাম, 'a.out'। এবং, আগে কম্পাইল করা কোনো 'a.out' থাকলে তার জায়গাতেই সেভ করে দেবে। তাই সচরাচর নিজের পছন্দের নাম দিয়ে দেওয়া হয়। ডিফল্ট নামটা কেন 'a.out' হয় তার একটা গল্প আছে, যাকগে। এই ডিরেক্টরিতে দাঁড়িয়ে বাইনারিটা চালাতে গেলে কমান্ড দিতে হবে, './gadha'। দিলেই স্ক্রিনে লেখা ফুটে উঠবে, উপরে নিচে এক লাইন ফাঁকা জায়গা রেখে, 'Kire Gadha!!!'। এবার এই গোটা কাজটা আমরা সি নামের প্রোগ্রামিং ল্যাংগুয়েজ দিয়ে না-করে ব্যাশ স্ক্রিপ্টিং ল্যাংগুয়েজ দিয়ে করতে চাইছি। প্রথমে স্ক্রিপ্টটা লেখা। একটা টেক্সট ফাইলে নিচের লাইনদুটো লিখে দিলাম।

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

ফাইলটা সেভ করলাম 'gadha.sh' নামে। '.sh' শেলস্ক্রিপ্টের প্রথাসিদ্ধ এক্সটেনশন, বলেছি আগেই। এর পরের স্টেপদুটোও আমাদের পরিচিত। এক নম্বর, টেক্সট ফাইলটাকে এক্সিকিউটেবল করা। নানা ভাবেই করা যায় এটা। 'chmod +x gadha.sh', বা, 'chmod 700 gadha.sh'। দু-নম্বর স্টেপ, স্ক্রিপ্টটা চালানো। তার কমান্ড, 'gadha.sh'-এর ডিরেক্টরিতে দাঁড়িয়ে './gadha.sh'। এবারেও ফলাফল সেই একই। স্ক্রিনে ফুটে উঠল আমাদের চেনা লাইনটা, উপরে নিচে এক লাইন করে ফাঁকা রেখে। অন্তত লাইনের নিরিখে এই স্ক্রিপ্টটা আগের সি-কোডটার চেয়ে বহু ছোট। এবং, কম্পাইল করার কোনো ঝামেলা নেই। তার চেয়েও বড় কথা, যে কোনো ওএস-এই চলবে, যদি সেখানে ব্যাশ থাকে। 'mplayer' বাইনারির মত, এই 'gadha' বাইনারিটাও কিন্তু অন্য ওএস-এ চলবে না, আলাদা করে কম্পাইল করতে হবে। প্রোগ্রামিং জগতের সমস্ত কাজ যদি মেশিনকে 'কিরে গাধা!!!' ডাকার কাছাকাছি সরলতার হত, তাহলে যে ব্যাশস্ক্রিপ্ট ছাড়া এই গ্রহে আর কোনো প্রোগ্রামিং মাধ্যম থাকত না, এটা নিঃসন্দেহ। অনেক বেশি জটিলতাকে অনেক সহজে ধরা যায় প্রোগ্রামিং ভাষাগুলোয়, আর সবচেয়ে বড় কথা ওই হার্ডওয়ার রসদের ব্যাপারটা। কিন্তু এ সত্ত্বেও, ব্যাশের নিজের কিছু জোরের জায়গা আছেই। এক, আলাদা কোনো সফটওয়ার চালাতে হয়না, গ্নু-লিনাক্স চলছে মানে পেটে পেটে ব্যাশও চলছে। তাই কম্পাইলার যে রসদটুকু নিত সেটা আর নতুন করে দিতে হবেনা এখানে। এমনকি রুবি বা পার্ল ইত্যাদি কোনো ইন্টারপ্রিটার মানেও কিছুটা আলাদা মেমরি তথা অন্য রসদ সে নেবেই। দুই, আগেই বলেছি, কোনো ব্যাশ স্ক্রিপ্ট চালানো মানে জাস্ট আর একটা ছানা-ব্যাশ, পুরোনো ব্যাশ প্রসেসেরই একটা চাইল্ড প্রসেস। তাই এই ছানা-ব্যাশ অনেকটা অব্দি মেমরি তথা অন্যান্য রসদ ভাগ করে নেবে ইতিমধ্যেই বাফারে থাকা ওই প্রারম্ভিক ব্যাশ প্রসেসের সঙ্গে, রসদের উপরে চাপ কম পড়বে। তিন, ব্যাশ আমাদের এমনিতেই চেনা, গ্নু-লিনাক্সে কাজ করা মানেই ব্যাশ শেখা, আলাদা করে শিখতে হবেনা। আমাদের এই পাঠমালার পরপর দিনগুলোকে ভাবুন, পাঁচ থেকে একে তো ব্যাশমালা বলাই যায়। কমান্ড দেওয়া মানেই তো ব্যাশকে কমান্ড দেওয়া। ফাইল কপি করা, নড়ানো, পাইপ করা, রিডাইরেক্ট করা ইত্যাদি যাই হোক। ব্যাশের গোটা ডকুমেন্টেশনটাই ইতিমধ্যেই সদাসর্বদাই রয়েছে আপনার সিস্টেমে। 'man bash' বা 'info bash' করুন এবং পড়ে যান। আর পাশের কনসোলে করে করে দেখতে থাকুন। এক্সউইন্ডোজ হলে একটা টার্মিনাল খুলে নিন।

এই 'gadha.sh' ব্যাশস্ক্রিপ্টটা আসলে কী? নিছক একটা কমান্ড যে নিজে আবার 'echo' নামের কমান্ডটা ব্যবহার করল। এর দ্বিতীয় লাইনটা কমান্ড প্রম্পটে লিখে এন্টার মারলে হুবহু একই কাজ হত। এর ব্যবহারের কমান্ডটাও সিস্টেমেই গোড়া থেকেই ছিল। ব্যাশস্ক্রিপ্ট ঠিক এটাই করে, সিস্টেমে থাকা এক বা একাধিক কমান্ডকে তাদের অপশন সহ বিভিন্ন ভ্যারিয়েবলের উপর পরপর 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 বড়, ব্যবহার করেছে চারটে কমান্ড। 'echo', 'read', 'cat' এবং 'expr'। তাদের অপশন সহ। ব্যবহার করেছে চারটে ভ্যারিয়েবলের উপর। লেখার ফাইলের নাম বা 'f', কাউন্টার বা 'c', বিভাজক বা 'd', এবং ডিরেক্টরির প্রতিটি ফাইল পরপর বা 'i'। আদেশগুলো সঠিক রকমে ব্যাশের কাছে পৌঁছে দেওয়ার জন্যে লেগেছে কিছু বিশেষ চিহ্ন। মন্তব্যবাচক চিহ্ন বা '#', রিডাইরেকশন চিহ্ন বা '>>', কমান্ড সাবস্টিটিউশনের চিহ্ন ব্যাককোট বা '`', অ্যাসাইনমেন্ট চিহ্ন বা '=', ভ্যারিয়েবলের মানবাচক চিহ্ন বা '$'। এরা এবং এই রকমের সবকিছুকে মিলিয়ে তৈরি হয় ব্যাশ নামের স্ক্রিপ্টিং ভাষা। অন্য প্রোগ্রামিং ভাষায় এই কমান্ডগুলো, বিশেষ শব্দগুলো,

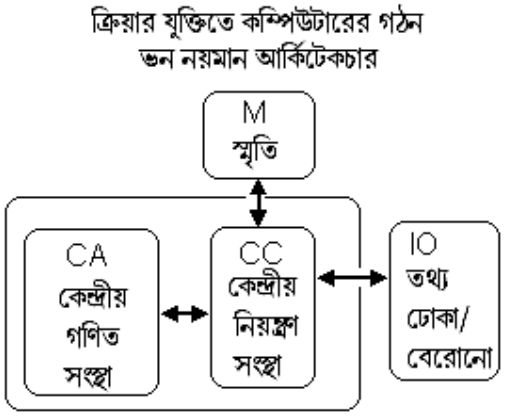
চিহ্নগুলো আলাদা করে শিখতে হয়, এখানে তা নয়, প্রতিমুহূর্তে একটা গ্নু-লিনাক্স সিস্টেমে কাজ করতে গিয়ে আমরা সেই কমান্ড শব্দ চিহ্নগুলো ব্যবহার করে চলি, এই পাঠমালাতেও করছি। এরা একই সঙ্গে ব্যাশ কমান্ড প্রম্পটের কমান্ড, আবার ব্যাশ ল্যাংগুয়েজেরও কমান্ড। ভ্যারিয়েবল নিয়ে এবং দু একটা চিহ্ন নিয়ে আলোচনা ইতিমধ্যেই হয়েছে। এবার আমরা ব্যাশ শেলের ভাষাগত এবং ক্রিয়াগত নিয়মকানুনগুলো আর একটু খুঁটিয়ে শিখব। যাকে আমরা বলব কন্ট্রোল স্ট্রাকচার বা নিয়ন্ত্রণ কাঠামো। শূন্য নম্বর দিনে কন্ট্রোল স্ট্রাকচারের আলোচনাটা মনে করুন, যা আবার তিন নম্বর দিনে আডার প্রসঙ্গে ফেরত এসেছিল। ক্যালকুলেটর থেকে সেই নাটকীয় উল্লম্ফন যা কম্পিউটারকে কম্পিউটার করেছিল — কন্ট্রোল স্ট্রাকচার বা নিয়ন্ত্রণ কাঠামো।

১০।। ভন নয়মান আর্কিটেকচার এবং নিয়ন্ত্রণ কাঠামো

শূন্য নম্বর দিনে ভন নয়মানের বেশি উল্লেখ করিনি ইচ্ছে করেই। স্টিফেন হকিংয়ের ব্রিফ হিস্ট্রি অফ টাইমের ভূমিকাতেই ছিল না, সমীকরণের সংখ্যার বর্গের ব্যাস্তানুপাতে কমে যায় পাঠকের সংখ্যা? খটোমটো নামের জন্যেও সেটা একই ভাবে সত্যি। আর সঠিক জায়গায় সঠিক নামত্যাগ করে (মলত্যাগের সঙ্গে যদি কোনো আত্মীয়তা পান, তা আমার বানানো না, ইংরিজিতে বলে নেম-ড্রপিং), সঠিক লোকের কাছে পৌঁছে দেওয়া — লাইনেই আছি দাদা, আত্মবিশ্বাসহীনতার এই কলোনিয়াল সংস্কৃতিটারও বাইরে যেতে চাইছিলাম। সাত নম্বর দিনে লিখেছিলাম, খুব চটে গিয়ে, নেম-ড্রপিং সিম্বল-ড্রপিং করে আবহটাকে ইচ্ছে করে আরো খটোমটো আরো এলিট করে দেওয়াটা একটা রাজনীতি। অনেলিট ঘরের কোনো টুম্পা যাতে আডা লাভলেস হওয়ার স্বপ্ন দেখার স্পর্ধাতেও পৌঁছতে না-পারে। লিনাক্স এইট করা লিনাক্স বিশারদ সিনিয়র সিটিজেন ধেড়ে পাঠারা যাতে সাহেব অ্যাকাডেমির সাহেব ড্রপিং ঘেঁটে যেতে পারে, আর বন্দুকের শব্দে বমকে যাওয়া স্যাভেজের বাচ্চারা তার সৌরভে আমোদিত হতে পারে — দশদিক থেকে যারা আসিতেছে চলে পিসি শিখবে পিসি শিখবে বলে। মার্জিন অফ মার্জিন বলে আমাদের একটা বইয়ে একটা শব্দ আছে 'savage', বানান লেখা হয়েছে 'sa**V**A**g**e'। প্রথম আর শেষটুকু মিলিয়ে হয় সেজ, সাধক, তপস্বী, প্রাজ্ঞ। আমি ব্যক্তিগতভাবে একজন পরিণামহীন ভবিষ্যতহীন মানুষ, জৈবনিক অর্থে। সমাজ বদলানোর স্বপ্ন, যাদের নিয়ে বড় হয়েছিলাম, নিহত হয়েছে চালু ইতিহাসের হাতে। 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 উত্তেজনাটা পেতে পারে — এর চেয়ে বড় পাওনা আর কী হয় — একটা স্যাভেজও যদি নিজের ভিতর সঙ্গোপন সেজ-সন্দর্শনে পৌঁছতে পারে, তার একটুও রসদ পায় আমার কাজ থেকে। এই গোটা বইটায় প্রতিমুহূর্তে আপনার ভাবাটাকে এগিয়ে দেওয়ার চেষ্টা করাটা অ্যাকাডেমির ধেড়ে পাঠাদের মহিমার বিরুদ্ধ রাজনীতির একটা ব্যক্তিগত যাত্রা।

শূন্য নম্বর দিনে কম্পিউটারের কাজের পাঁচটা উপাদানের কথা বলেছিলাম। ১, তথ্য ঢোকানোর ইনপুট ডিভাইস। ২, তথ্য বার করার আউটপুট ডিভাইস। ৩, তথ্য চটকানোর প্রসেসিং ডিভাইস। ৪, তথ্য ধারণের হোল্ডিং ডিভাইস। ৫, সবচেয়ে গুরুত্বপূর্ণ, ১ থেকে ৪ এই চারটে কাজের নিয়ন্ত্রণের কন্ট্রোল ডিভাইস। ছবিতে ভন নয়মানের মূল ছকটার সঙ্গে মেলান এদের। ভন নয়মানের ছকে ইনপুট আর আউটপুট উপাদান দুটোকে একটা উপাদান হিশেবে ধরা হয়েছে — আইও বা ইনপুট/আউটপুট। কম্পিউটার বিজ্ঞানে ভন নয়মানের অবদানের কথা আমরা আগেই বলেছি, চার নম্বর দিনে। পেনসিলভানিয়া বিশ্ববিদ্যালয়ে থাকাকালীন এডভ্যাক (EDVAC) মেশিনের সূত্রে ১৯৪৫ সাল নাগাদ নয়মান স্টোরড-প্রোগ্রাম-কম্পিউটারের লজিকাল গঠনের যে ছকটা দিয়েছিলেন, সেটা এখানে তুলে দিচ্ছি। 'স্টোরড-প্রোগ্রাম' বিষয়টা খেয়াল করুন। এর মানে, প্রোগ্রাম বা আদেশ-তথ্যটাকে স্মৃতিতে তুলে কাজ করছে কম্পিউটার। 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' করে গাণিতিক হিশেবের কাজ, চারটে মূল পাটীগাণিতিক ক্রিয়া, যোগ বিয়োগ গুণ ভাগ, এবং তাদের মিলিয়ে আরো কিছু, যেমন সূচক বা ইন্ডেক্স, রুট বা মূল, বা লগ এবং ত্রিকোনমিতিক ফাংশনের হিশেবের কাজ ইত্যাদি। 'M' তথ্য ধরে রাখে, হিশেবের কাজের গাণিতিক তথ্য মানে বিভিন্ন সংখ্যা, এবং কী ভাবে কী কাজ করতে হবে সেই নির্দেশের তথ্যটাও (নির্দেশ-তথ্য মানে এখনকার কম্পিউটারের নিরিখে কম্পাইলড বাইনারি বা ইন্টারপ্রিটেড স্ক্রিপ্ট-কোড)। ইউজারের সঙ্গে মেশিনের পারস্পরিক সম্পর্কের ইন্টারফেস

'IO', কাঁচামাল তথ্য মেশিনে ঢোকানো এবং উৎপাদিত তথ্য মেশিন থেকে বার করার। 'CC' নিয়ন্ত্রণ করে অন্য সমস্ত কাজদের — কোন তথ্যের উপর কোন নির্দেশ কাজ করবে, কতদূর অব্দি কী কাজ করবে, কখন শুরু কখন শেষ হবে, কখন কোন কাজ বদলাবে, ইত্যাদি পুরোটা।



ভন নয়মানের এই ছকটা একটা লজিকাল বা যুক্তিবৈজ্ঞানিক ছক। মেশিন বা যন্ত্রাংশের ভৌত গঠন নিয়ে মাথা ঘামাননি নয়মান। কম্পিউটার-প্রক্রিয়ার যুক্তিগত প্রবাহটাকে ধরতে চেয়েছিলেন। ছকটাকে আর এক ভাবে ভাবুন। ধরুন, ক্যালকুলেটরে হিশেব কষছেন। যে কাগজে রাফ লিখছেন সেটা অবশ্যই 'M', তার কিছুটা অবশ্য ভালো ক্যালকুলেটরে সার্কিটেও থাকে। ক্যালকুলেটরের কি আর ডিসপ্লে মিলিয়ে 'IO'। ভিতরের সার্কিটটা 'CA'। কিন্তু 'CC' কই? ক্যালকুলেটরে হিশেব কষার সময়ে 'CC' আপনি নিজে, সচেতন নিয়ন্ত্রক সংস্থা। ক্যালকুলেটরটা জড় পদার্থ বলেই একটা সচেতন নিয়ন্ত্রক সংস্থার প্রয়োজন পড়ছে। আর স্টোরড-প্রোগ্রাম-কম্পিউটারে সেই নিয়ন্ত্রণটাকে বকলমা দেওয়া হচ্ছে মেশিনের মধ্যে ধরে রাখা ওই প্রোগ্রামে। নিজের মাথার ভার কিছুটা সময়ের জন্যে মানুষ দিয়ে দিচ্ছে মেশিনকে। প্রোগ্রামিং ভাষার কোড/কম্পাইলার/বাইনারির হাতে, ব্যাশ-স্ক্রিপ্টিং-এ ব্যাশ-স্ক্রিপ্ট/ব্যাশের হাতে। এরাই হয়ে দাঁড়াচ্ছে সচেতন নিয়ন্ত্রকের প্রতিনিধি। এবার ব্যাশ নামক স্ক্রিপ্টিং ভাষার ব্যাকরণটা একটু জানতে হবে, জ্যান্ত মানুষের সচেতন নিয়ন্ত্রণকে সে ঠিক কী ভাবে বাস্তবে প্রয়োগ করে সেটা বুঝতে।

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 কর্মকাণ্ডের। কর্মকাণ্ডটা আসলে কী? শূন্য থেকে দশ, অনেকটা পেরিয়ে এলাম, আজ এগারো নম্বর দিন, এর গোটাটা মাথায় রেখে ভাবুন। শূন্যয় বলেছিলাম, শেষ অব্দি কম্পিউটারের কাজকর্ম খুব সরল সাধাসিধে, ডিরেক্ট। কোনো ধূসরতার সন্ধ্যাভাষার ধুন্ধলে শাহি উপাখ্যান সেখানে ছুপা নেই। গোটাটাই বাইনারি, শূন্য আর এক — এই হল কম্পিউটারের তথ্য মানে ডেটা। আমার মেশিনের দুটো চল্লিশ জিবি করে হার্ডডিস্ক '/dev/hda' আর '/dev/hdb'-র কথা বলেছি, মোট আশি জিবি। এই আশি জিবি ডিস্কভূমির মোট ব্যবহৃত এই মুহূর্তে ৩২.৫ জিবি। পেলাম সবগুলো পার্টিশনে মাউন্ট করার পর '`df -h`' কমান্ড দিয়ে, প্রতিটি পার্টিশনের ব্যবহৃত ভূমিগুলো যোগ করে। ৩২.৫ জিবি মানে ৩২.৫×১০২৪×১০২৪×১০২৪×৮ বিট, বা, ২.৭৯১৭×১০^১১ বিট, মানে, মোটামুটি ধরুন ২৮-এর পরে দশটা শূন্য দিলে যে সংখ্যাটা হয়, ততগুলো খোপ, প্রতিটি খোপে আছে হয় ১ নয় ০ — এইটা হল আমার মেশিনে এই মুহূর্তে বসবাসমান মোট তথ্যের পরিমাণ। আটাশ হাজার কোটি কুচো কুচো চৌম্বক খোপে হয় ১ নয় ০। খোপের সংখ্যাটাকে বাড়িয়ে তোলা যাবে, আর হার্ডডিস্ক না বাড়িয়ে, ৬.৮৭১৯৫×১০^১১ বা মোটামুটি হিশেবে উনসত্তর হাজার কোটি অব্দি। মোট হার্ডডিস্কভূমিতে মোট যত শূন্য বা এক লেখা যায়, সেটা যদি হাতে করে লিখতে হত, আর মিনিটে যদি একশোটা ০ বা ১ লিখতে পারি, পোস্টচল্লিশে এসে আর কত স্পিডে লিখব, আমার মাত্র ১৩০৭৪ বছর লাগত। এক বছরের মধ্যে গোটাটা লিখতে হলে, নিজে লেখার সঙ্গে, তেরো হাজার

তিহাত্তর জন সহকারী রাখতে হত। তিন নম্বর দিনে ছিল, হিশেব করতে করতে মানুষের মনে হল, অ্যাবাকাসকে একটু সহকারী রাখার। সেই একটু সহকারী মানে মাত্তর তেরো হাজার তিহাত্তর। তাও এটা শুধু লেখার। ব্যারন দ প্রনির হিউম্যান কম্পিউটারদের মোট কতজনের হিশেবের কাজ করে এক একটা মেশিন, সেই হিশেব নয় এটা।

কাজের কথায় আসা যাক। এই আমার মেশিনের এই আটাশ হাজার কোটি চৌম্বক খোপে কী আছে? আছে তথ্য। দুরকম তথ্য। এক, যে তথ্যকে চটকাতে হবে, ধরুন তার নাম দিচ্ছি কাঁচা বা র তথ্য। আর, এই কাঁচা তথ্যকে কী ভাবে চটকাতে হবে, সেই আদেশ মালাটাও তথ্য, কম্পিউটারের নাড়িভুঁড়িতে সেটাও ভরা আছে এই আটাশ হাজার কোটি খোপের শূন্য আর এক দিয়েই। কাঁচা তথ্যের বাইরে এই নির্দেশ তথ্য, ইন্সট্রাকশন ডেটা, সেটাও তথ্য, একেই নয়মান 'স্টোরড প্রোগ্রাম' বলে ডেকেছিলেন। তখনো প্রোগ্রামিং এই ভাবে গড়ে ওঠেনি, এখন আমাদের কাছে এই প্রোগ্রাম মানে নানা জাতের নানা কিসিমের বাইনারি। কেউ অডিও প্যাকেজ, সাঙ্গীতিক তথ্য চটকায়, কেউ ওয়ার্ড প্রসেসর, শুধু লিখিত শব্দ চটকায়, কেউ ব্রাউজার, মেশিন থেকে মেশিনে ভ্রাম্যমান তথ্য চটকায়, এবং ইত্যাদি। তাহলে কম্পিউটারের মোট ইতিহাস এবং ভূগোল কী দাঁড়াল? পেটের মধ্যে ভরা কোটি কোটি ০ বা ১ লেখার খোপ — এটা তার ভাণ্ডার। এবং তার কাজ কী? এই কোটি কোটি ০ বা ১ ইধার-কা-মাল-উধার করা। গাণিতিক প্রতিতুলনা ব্যবহার করা যাক। এই অগণ্য ০ বা ১, এরা থিতু হয়ে আছে, তখন এরা তথ্য, তথ্য তখন স্ক্যালার। আর, সেই ০ বা ১ এরা নড়ছে, এক ঠাঁই থেকে আর এক ঠাঁই যাচ্ছে, এক আকার থেকে আর এক আকার, এর মানে এদের চটকানো হচ্ছে, মানে মেশিন কাজ করছে, প্রোগ্রাম রান করছে। তথ্য তখন ভেক্টর। তার একটা গতি আছে, দিকনির্দেশ আছে।

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

এক নম্বর দিনের সেই ওল্টানো সৌধের মত হার্ডওয়ার থেকে সফটওয়ারের হায়েরার্কির ছকটা আর একবার দেখুন মন দিয়ে। একদম নিচের স্তরে, হার্ডওয়ারের স্তরে ভৌত চৌম্বকতার শূন্য আর এক, এবং আরো অন্যান্য ভৌত উপাদান, এই নিয়ে স্তর ১। এর উপরে মাইক্রোআর্কিটেকচারের স্তর ২-এর ভিত, যেখানে দাঁড়িয়ে আছে মেশিন ল্যাংগুয়েজ স্তর ৩। এই তিনটেকে মিলিয়ে হার্ডওয়ার বা ভৌত যন্ত্রপাতির এলাকা। এবার মূল কাঠামো বা সিস্টেম প্রোগ্রামের এলাকা শুরু। তার মধ্যে দুটো স্তর, নিচের ভিতটা হল অপারেটিং সিস্টেম স্তর ৪ মানে আমাদের গ্নু-লিনাক্স কারনেল। তার উপরে তিনটে সমান্তরাল জিনিষ, কম্পাইলার এডিটর কমান্ড-ইন্টারপ্রিটার মিলিয়ে স্তর ৫। এখন এদের চিনি, কম্পাইলার মানে জিসিসি, এডিটর মানে ইম্যাক্স ভিম জাতের কেউ, আর কমান্ড-ইন্টারপ্রিটার, সেটা কে, জানেন নাকি? এখানেই শেষ সিস্টেম প্রোগ্রামের এলাকা। এর উপরে প্রায়োগিক সফটওয়ারগুলোর স্তর ৬, প্রতি মুহূর্তে যাদের নিয়ে আমরা কাজ করছি। নানা কাজের নানা বাইনারি বা বাইনারি-সমাহার।

তাহলে দেখুন, ভৌত চৌম্বকতার ০ বা ১ দিয়ে তথ্যকে দেখাটা থেমে যাচ্ছে হার্ডওয়ারের স্তরে। এর পর থেকে মেশিনে তথ্য মানে সরাসরি যেসব সংখ্যা বা চিহ্ন নিয়ে আমরা কাজ করি। কিবোর্ডে টাইপ করে যে বর্ণসমাহার পাই, এবং নিউলাইন, ক্যারেজ রিটার্ন, বেল ইত্যাদি আরো কিছু চিহ্ন। এদের মিলিয়ে যেটা তৈরি হচ্ছে সেটা আমাদের কাছে বোধ্য, হাইলেভেল হিউম্যান আন্ডারস্ট্যান্ডেবল ভাষার সংখ্যা এবং চিহ্ন দিয়ে তৈরি তথ্য। এডিটর কম্পাইলার এবং ব্যাশ ইত্যাদি কমান্ড ইন্টারপ্রিটারে আমরা যেভাবে কাজ করি। আমাদের পড়ার এবং বোঝার মত চিহ্নে তথা ভাষায় তৈরি বলেই আমরা তাদের নিয়ে কাজ করতে পারি। শূন্য নম্বর দিনের তথ্যের পরিমাপ এবং অ্যাসকি সংক্রান্ত

আলোচনাটা মনে করুন। কিন্তু এই কম্পাইলার, এডিটর বা কমান্ড ইন্টারপ্রিটারের মূল বাইনারি বা এদের উপরের স্তরে প্রয়োগ-সফটওয়ারের কোনো বাইনারি খুলে দেখুন, সেটা সম্পূর্ণ অবোধ্য। শূন্য নম্বর দিনে একটু ভিশুয়াল চ্যাংড়ামি করেছিলাম, মনে পড়ছে পিডিএফ ফাইল সরাসরি এডিটর দিয়ে খুলে পাওয়া ক্রিপ্টন গ্রহের সংখ্যাগুরু জনগোষ্ঠীর মাতৃভাষায় লেখা কাব্যগ্রন্থের কথা? অথচ তাকেই পাঠযোগ্য করে খুলে দিচ্ছে এক্সপিডিএফ গোস্টভিউ ইত্যাদি পিডিএফ পড়ার বাইনারিগুলো। তাই, আমাদের কাছে বোধ্য হোক, বা অবোধ্য, সমস্ত তথ্যই হল চিহ্ন দিয়ে তৈরি, চিহ্ন সমাহার। এই চিহ্ন সমাহার একটা প্রবাহের মত। নিষ্ক্রিয় মেশিনে স্থির হয়ে আছে চিহ্নমালা, সেটাই তথ্য, কাঁচা তথ্য বা নির্দেশ তথ্য, র ডেটা বা প্রোগ্রাম। যখন মেশিন চলছে, প্রোগ্রামগুলো রান করছে, তখন সেই চিহ্নপ্রবাহ নড়ছে, একটা নির্দিষ্ট গতিমুখে নির্দিষ্ট নিয়ম মেনে নড়ছে, যে নিয়মগুলো ওই প্রোগ্রামে লেখা আছে।

তার মানে কী দাঁড়াল? কম্পিউটারে শেষ আছে কেবল 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, প্রোগ্রামগুলো চলাকালীন, চারিদিকে প্রলয়নৃত্য-পরায়ন নটরাজের মুদ্রার মত বহুমুখী বিচিত্রমুখী নিরবচ্ছিন্ন চিহ্নপ্রবাহের ট্র্যাফিক সে সামলাতে পারবেনা। ঠিক এটাই করে ব্যাশ, 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 ব্যাশের ক্রিয়াকাণ্ডের দিকনির্দেশের তিনটে গুরুত্বপূর্ণ ভেক্টর এই তিন জন। স্ট্যান্ডার্ড ইন, 'Std. In', স্ট্যান্ডার্ড আউট, 'Std. Out', এবং স্ট্যান্ডার্ড এরর, 'Std. Err'। পাঠমালা জুড়ে বারবার এদের উল্লেখ করছি, এমনকি না-জেনেই ব্যবহারও করছি, রিডাইরেকশন বা পাইপ-এ, কিন্তু ঠিক ভাবে কখনো আসছে না। এবারে আসবে তারা তাদের পুরোটা নিয়ে। আগে বহুবার বলেছি, গ্নু-লিনাক্সে সবই এক একটা ফাইল। স্ট্যান্ডার্ড ইন, স্ট্যান্ডার্ড আউট এবং স্ট্যান্ডার্ড এরর — এরাও তিনটে 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। তিন রকম বাইট স্রোতের জন্যে তিনটে ফাইল। যে ফাইলগুলোকে ব্যাশ তথা ব্যাশাধীন সমস্ত কমান্ড নিজের ইনপুট বা আউটপুট হিশেবে ব্যবহার করে।

সঠিক টেকনিকাল অর্থে বললে, ব্যাশ কোনো কমান্ডই কিবোর্ড থেকে নেয়না, নেয় স্ট্যান্ডার্ড ইন থেকে। ডিফল্ট সেটিং-এ স্ট্যান্ডার্ড ইন হল আপনার কিবোর্ড। একই ভাবে বললে, আপনার কোনো কমান্ডের ফলাফলই ব্যাশ স্ক্রিনে পাঠায় না, পাঠায় স্ট্যান্ডার্ড আউটে, যা ডিফল্ট সেটিং-এ আপনার স্ক্রিন। আর, কমান্ড 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। পাঠমালায় বারবার আমরা 'cat' করে '.bashrc' ইত্যাদি ফাইলকে অন্য কোনো ফাইলে চালান করেছি। তার জন্যে আমরা যে কমান্ড দিয়েছি সেটা এরকম, '`cat ~/.bashrc > bashrc.text`'। এবার তার বদলে দেওয়া যাক, '`cat - ~/.bashrc - ~/.profile > local.conf.text`'। এবার এন্টার মারুন। স্ক্রিনে দেখুন, 'cat' কমান্ড দেওয়ার পরে যেরকম প্রম্পটহীন কালো স্ক্রিন আসে, সেইরকম 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, যাতে একটা থেকে অন্যটাকে আলাদা করা যায়। '-' চিহ্নটা ঠিক এটাই করবে। কালো স্ক্রিনে টাইপ করে যা যা দেবেন, সেই তথ্য এবং ফাইলগুলো ব্যাশ পরপর রাখতে রাখতে যাবে 'local.conf.text' ফাইলে। প্রথমে টাইপ করুন, '`file: '~/.bashrc'`'। এরপর একটা আস্ত লাইন পরপর '=' চিহ্ন টাইপ করে দিন, এই দাগটা হেডিং থেকে ফাইলকে আলাদা করবে। এবার পরপর দুবার এন্টার মারুন, প্রথম এন্টারে পরের ফাঁকা লাইনে গেল, আর দ্বিতীয় এন্টার যাতে একটা লাইন ফাঁকা রাখে '.bashrc' ফাইলের আগে। এবার 'Ctrl-D' মারুন। এমনিতে 'cat' করার সময় এতেই কমান্ড প্রম্পট ফেরত আসে, মানে, ক্যাট করার কাজ শেষ হল। এখানে তা হয়নি, এখনো সেই কালো স্ক্রিন। এখন আসলে দ্বিতীয় '-' চিহ্নটার আদেশ পালন করছে ব্যাশ। প্রথমে দুটো এন্টার মারুন, এক লাইন ফাঁকা জায়গা বানানোর জন্যে। তারপর আগের মত একই ভাবে একটা

হেডিং টাইপ করুন, '.profile' ফাইলের জন্যে, '`file: '~/.profile'`', আবার এক লাইন '=' চিহ্ন। আবার দুটো এন্টার। 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। একবার '`cat local.conf.text`' করে দেখুন, ঠিক যেভাবে আমরা দিয়েছি, সেভাবেই এসেছে কিনা। অবশ্যই এসেছে। ব্যাশ কখনো বেআদবি করেনা। অর্থাৎ, যেমন 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। এই শেষটা আমরা এখনি দেখব।

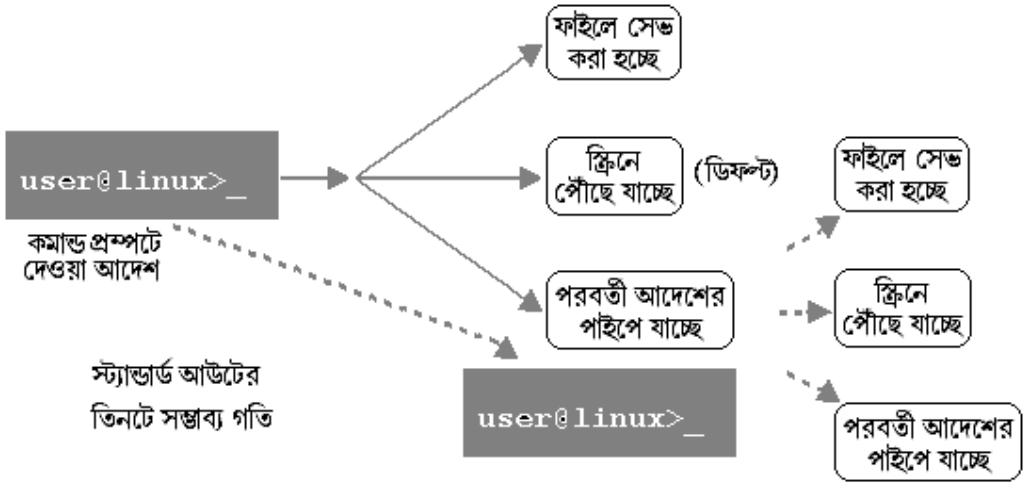
একটা কমান্ডকে অন্য কমান্ডের আউটপুট থেকে নিজের ইনপুট নিতে আগেও দেখেছি। ছয় নম্বর দিনে আমরা যখন '`man -k CD | less`' করে 'CD' শব্দটা কোন কোন ম্যানুয়াল পেজে আছে সেই তালিকাটা পড়ছিলাম, বা '`man -k CD | grep 'audio'`' করে সেই তালিকাটার কোন কোন লাইনে 'audio' শব্দটা আছে সেটা খুঁজে বার করছিলাম, তখন কী ঘটছিল সেটা বোঝার চেষ্টা করুন। 'less' নামের পেজারটা একটা কমান্ড, পাতার এককে একটা চিহ্নপ্রবাহ বা ক্যারেকটার স্ট্রিমকে দেখায়। '`less`' কমান্ডটা 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 করবে '`grep`', যেখানে উদ্দীষ্ট 'audio' শব্দটা আছে। এখানেও দেখুন, স্ক্রিন বা 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। এরা এদের আউটপুটকে স্ট্যান্ডার্ড আউটে না-দিয়ে একটা ডিস্ক ফাইলেও দিতে পারত। ওখানেই, '`man -k CD | grep 'audio'> audio.cd.program.text`' কমান্ড দিয়ে আমরা 'man' হয়ে 'grep'-এর ফলাফলটাকে 'audio.cd.program.text' ফাইলে রিডাইরেক্ট করেছিলাম।

অর্থাৎ, একটা কমান্ডের কিছু এসে যায়না, তার ইনপুট কোথা থেকে আসছে, স্ট্যান্ডার্ড ইন না কোনো ডিস্ক ফাইল। বা, তার আউটপুট কোথায় যাচ্ছে, স্ট্যান্ডার্ড আউট না কোনো ডিস্ক ফাইল, তাতেও তার কিছু এসে যায়না।স্ট্যান্ডার্ড ইন, স্ট্যান্ডার্ড আউট, স্ট্যান্ডার্ড এরর — এরা প্রত্যেকেই এক একটা ফাইল, ক্যারেকটার স্ট্রিম বা চিহ্নের প্রবাহ। প্রথম ফাইলের ডিফল্ট উৎস কিবোর্ড, আর পরের দুটোর ডিফল্ট মোক্ষ স্ক্রিন বা টার্মিনাল। পরের দুটো একই ক্ষুরে মাথা মোড়াবে, এভাবেই ব্যবস্থা করা। এইমাত্র দেখলাম, এখুনি আরো ভালো করে বুঝব, কিছু বিশেষ চিহ্ন আর আদেশ আছে, যাদের কাউকে পাওয়ামাত্র ব্যাশ কমান্ড বা আদেশগুলোর সঙ্গে জড়িত চিহ্নপ্রবাহের ডিফল্ট উৎস বা মোক্ষ বদলে ফেলে। কিছু চিহ্ন বা কায়দা আগেই দেখেছি। যেমন '>' বা '<' বা '|' ইত্যাদি। এই বদলে ফেলাটাই হল রিডাইরেকশন বা চালান করা। রিডাইরেকশন যখন ঘটে একটা কমান্ড থেকে আর একটা কমান্ডে, তখন আমরা বলি প্রথম কমান্ডের আউটপুট বা ফলাফলকে দ্বিতীয় কমান্ডে পাইপ করা হল বা পথ-দেখানো হল। এই 'চালান' এবং 'পথ-দেখানো' প্রতিশব্দদুটো আমরা এনেছিলাম পাঁচ নম্বর দিনে। ভালো না, যাকগে, মানে তো বোঝা যাচ্ছে।

১২।। চিহ্নপ্রবাহের গতিপথের চালচিত্র

একটা আদেশ থেকে তৈরি হয় কিছু ফলাফল, কিছু চিহ্নপ্রবাহ বা ক্যারেকটার-স্ট্রিম, যা একটা ফাইল, তার নাম স্ট্যান্ডার্ড আউটপুট। এই স্ট্যান্ডার্ড আউট নামের চিহ্নপ্রবাহের ফাইলটার ভবিষ্যত হতে পারে তিন রকম। এক, তার ডিফল্ট গতি স্ক্রিন। কমান্ডটা সরাসরি আপনার চোখের সামনে স্ক্রিনে এনে দিল ফলাফলটাকে। দুই, এই কমান্ডের ফলাফল বা আউটপুটটা চালান করে দেওয়া হল একটা ডিস্কফাইলে, আপনার ডিস্কভূমিতে বিশেষ একটা ফাইলে আউটপুটটা সঞ্চিত হল, সেভ করা হল। তৃতীয় গতিটা হল ত্রিশঙ্কু, মানে, এই কমান্ডের সরাসরি কোনো আউটপুট তৈরি হলনা। এই কমান্ডের আউটপুটটা পাইপ হয়ে গেল আর একটা কমান্ডের কাছে, সেই কমান্ডের ইনপুট হয়ে। এই দ্বিতীয় কমান্ডটাও আপনি দিয়েছেন ওই কমান্ড প্রম্পট থেকেই। দ্বিতীয় কমান্ডটা এবার প্রথম কমান্ডের

আউটপুটটাকে নিজের মত করে প্রসেস করবে। অর্থাৎ, প্রথম কমান্ডের ফলাফল থেকে তৈরি চিহ্নপ্রবাহের স্ট্যান্ডার্ড আউট এবার দ্বিতীয় তথা পরবর্তী কমান্ডের ইনপুট হয়ে গেল। একই ইতিহাস এবার দ্বিতীয় কমান্ডেরও। প্রসেস করা শেষ হওয়ার পর, আবার সেই একই তিনটে সম্ভাব্য গতি। এক স্ক্রিন, দুই ফাইল, তিন পুনঃপাইপিত হওয়া। এই গোটা প্রক্রিয়াটাকে দেখুন, এই ছবিতে দেখানোর চেষ্টা করেছি।

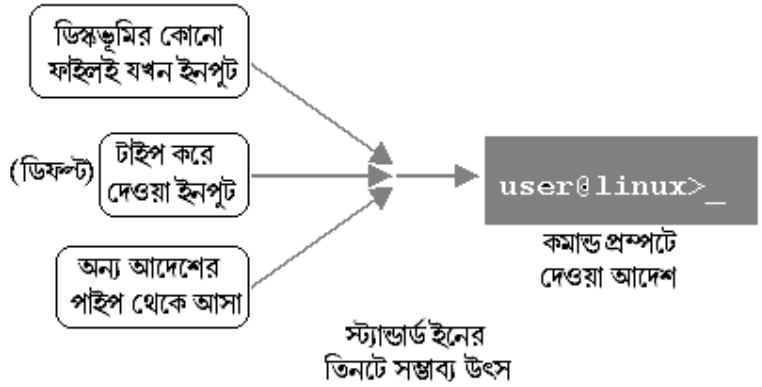


এখানে দেখুন, প্রথম কমান্ডের কাজকে দেখিয়েছি স্বাভাবিক দাগে, দ্বিতীয় কমান্ডের কাজটাকে দেখিয়েছি সচ্ছিদ্র দাগে, ডটেড লাইনে। আর দুটো কমান্ড প্রম্পটের মধ্যে যোগটাও দেখানো হয়েছে সচ্ছিদ্র দাগে। এটা বোঝাতে যে, আসলে, কমান্ড প্রম্পট কিন্তু দুবার না, একবারই মাত্র ব্যবহার হয়েছে। সেই একবার ব্যবহারেই ব্যাশের কাছে পৌঁছে দেওয়া হয়েছিল দুটো কমান্ডকেই। একটু আগে আমাদের দেওয়া রিডাইরেকশনের কমান্ডটাই মেলান ছবির সঙ্গে, '`man -k CD | grep 'audio'> audio.cd.program.text`'। 'man' এবং 'grep' দুটো আদেশই দেওয়া হয়েছিল একইসঙ্গে, একই কমান্ড লাইনে। প্রথম কমান্ড 'man' তার ফলাফল পাঠিয়ে দিয়েছে দ্বিতীয় কমান্ড 'grep'-এর কাছে। এটাই পাইপ করা বা পথ-দেখানো, যা ব্যাশকে বোঝানো হয়েছে '|' চিহ্ন দিয়ে। দ্বিতীয় কমান্ড 'grep' তার নিজের ফলাফলটাকে পাঠিয়েছে 'audio.cd.program.text' নামে ফাইলের কাছে, যা এই ছবিতে একদম উপরের সম্ভাবনাটা। এটা হল রিডাইরেক্ট বা চালান করা, যার চিহ্ন ছিল '>'। ব্যাশক্রিয়া এখানেই শেষ। এর পরে আরো কমান্ড লাগানো যেত, '`man -k CD|grep 'audio'|less`' জাতীয় কমান্ডে। যখন 'grep'-এর ফলাফলটাকে ফাইলে না-পাঠিয়ে এবার পাইপ করা হল আর একটা তৃতীয় 'less' কমান্ডে । 'less'-এর কাছে সেটা ইনপুট। 'less' এবার এই চিহ্নপ্রবাহটাকে নিজের মত করে প্রসেস করে পৌঁছে দিল স্ক্রিনে, পাতা বাই পাতা তাকে দেখা গেল। তার মানে এবার, এতক্ষণে, মধ্যে তিনটে কমান্ডের পথ পেরিয়ে, দুটো কমান্ডের পাইপ পেরিয়ে, স্ট্যান্ডার্ড আউট তার ডিফল্ট গতিপ্রাপ্তিতে পৌঁছল। এবার পরের ছবি, স্ট্যান্ডার্ড ইন।

স্ট্যান্ডার্ড আউটপুটের পৌঁছনোর জায়গা বা টার্মিনাসের সংখ্যা তিন। স্ট্যান্ডার্ড ইনপুটের উৎসের সংখ্যা তিন। আগের স্ট্যান্ডার্ড আউটের ছবিতে তীরচিহ্নের দিকনির্দেশটা খেয়াল করুন। কমান্ড থেকে স্ট্যান্ডার্ড আউটের দিকে চিহ্নের প্রবাহকে বোঝাচ্ছে। পরের স্ট্যান্ডার্ড ইনের ছবিতে দেখুন, এখানে তীরচিহ্নটা উল্টো দিকে, চিহ্নের প্রবাহটা স্ট্যান্ডার্ড ইন থেকে কমান্ডে যাচ্ছে। কমান্ড প্রম্পটে দেওয়া কমান্ড তার ইনপুট হিশেবে যে চিহ্নপ্রবাহ বা ক্যারেকটার স্ট্রিমকে ব্যবহার করে, তার নাম স্ট্যান্ডার্ড ইন। কমান্ডের কাছে কাছে এই স্ট্যান্ডার্ড ইন আসার ডিফল্ট উপায়টা হল কিবোর্ড, কিবোর্ড থেকে টাইপ করে দেওয়া ইনপুট। কিন্তু এছাড়াও আসতে পারে। আসতে পারে ডিস্কভূমিতে ইতিমধ্যেই সেভ করে রাখা কোনো ফাইল থেকে। আসতে পারে অন্য কোনো আদেশের পাইপ থেকেও। পূর্ববর্তী সমাধিত কমান্ডের আউটপুটটা তখন এই কমান্ডের ইনপুট। আগের '`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?

যদি কমান্ড দিতাম, ‘`man -k CD > cdtemp; grep 'audio' cdtemp|less`’, তাহলে কী হত? প্রথমে ‘man’ কমান্ডের ফলাফলটা চালান হচ্ছে ‘cdtemp’ নামের ডিস্কফাইলে। ডিস্কফাইল বলতে বোঝাচ্ছি ডিস্কবাসী ফাইল। গ্নু-লিনাক্সে সবই ফাইল। এখানে যা নিয়ে চর্চা হচ্ছে সেই ক্যারেকটার স্ট্রিম বা চিহ্নপ্রবাহ, সেটাও ফাইল। কিন্তু সেই ফাইল আমাদের চেনা প্রাত্যহিক ফাইল নয়। এই ফাইলের আবাস সাইবার প্রক্রিয়ার ভৌতিক বা ভারচুয়াল 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 চেনা ফাইল হয়ে ওঠে। ডিস্কফাইল নাম দিয়ে এটাই বোঝাচ্ছি যে এটা ওই ধরনের কোনো সাইবারবাসী ঘনচক্কর ফাইল নয়, নিতান্তই ভাতডাল খাওয়া রোজকার ফাইল, এমনকি পকেট হাতড়ালে বনগাঁ লাইনের মান্থলিও বেরোতে পারে।

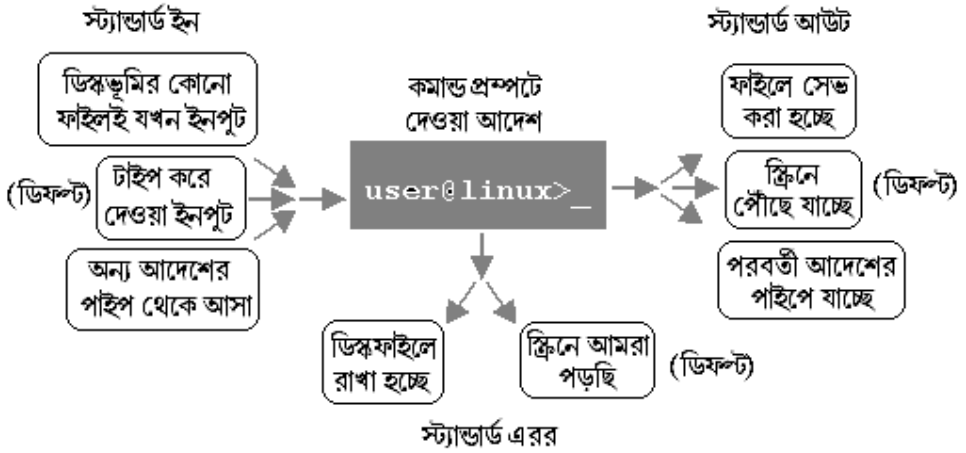
হ্যাঁ, যা বলছিলাম, আগের কমান্ডটায়, ‘man’ কমান্ডের ফলাফলটা স্ট্যান্ডার্ড আউটের ডিফল্ট স্ক্রিনে না গিয়ে চালান হচ্ছে ‘cdtemp’ নামের ডিস্কফাইলে। এর পরে দেখুন একটা ‘;’ চিহ্ন, সেমিকোলন দিয়ে ব্যাশের কাছে দুটো আলাদা কমান্ড পৌঁছে দেওয়ার কায়দাটা আগেই বলেছি। আগের কমান্ডটা আসলে পরপর দুটো কমান্ডের একটা সমাহার। পরপর কাজ করার দুটো কমান্ডকে আলাদা করেছে ‘;’ চিহ্ন। এবার পরের অংশের কমান্ডটুকুকে আলাদা করে বোঝার চেষ্টা করুন, ‘`grep 'audio' cdtemp | less`’। আগে ‘grep’ তার ইনপুট পাচ্ছিল ‘man’ কমান্ডের ফলাফল থেকে। এখন ‘grep’ তার ইনপুট নিচ্ছে ‘cdtemp’ নামের ডিস্কফাইল থেকে। সেই ফাইলের মধ্যে ‘audio’ শব্দটা কোন কোন লাইনে আছে সেটা খুঁজে দেখছে। সেই লাইনগুলোকে ‘grep’ সামনে স্ক্রিনে, মানে আউটপুটে তুলে আনতেই পারত। আনছে না। ‘grep’-এর আউটপুট পাইপ হয়ে যাচ্ছে ‘less’-এর কাছে। আমরা তাকে ‘less’ দিয়ে পড়ছি। তার মানে, ‘grep’ কমান্ডকে ইনপুট হিশেবে যাই দিই — সে ‘man’ কমান্ডের ফলাফলের পাইপ হোক, বা সেই ফলাফল দিয়ে গজিয়ে তোলা ডিস্কফাইল ‘cdtemp’ হোক, তাতে ‘grep’-এর কিছু এসে যায়না। আবার, নিজের কাজের ফলাফলকে ‘grep’ কোথায় দিচ্ছে, স্ক্রিনের স্ট্যান্ডার্ড আউটে, না ‘less’-এর ইনপুটে তাতেও কিছু এসে যায়না ‘grep’-এর।



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। সেখানে আমরা একটা 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।

স্ট্যান্ডার্ড ইন আর স্ট্যান্ডার্ড আউটেই ব্যাশচরিত শেষ হয়না। সবসময় সব আদেশ ঠিক ভাবে পালন করা হয়ে ওঠেনা ব্যাশের। আদেশ পালন করতে গিয়ে বহু রকম প্রমাদ ঘটে। ধরুন, এমন একটা ফাইল আপনি কনক্যাটেনেট বা '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, নামেই বোঝা যাচ্ছে, সেটা একটা অস্থায়ী বা টেম্পোরারি ফাইল। কাজ মেটার পর উড়িয়ে দিয়েছি। হঠাৎ যদি কমান্ড দিই, '`cat cdtemp`', স্বাভাবিকভাবেই, ব্যাশ সেই নামের কোনো ফাইল খুঁজে পাবেনা, ফাইলটাই তো আর নেই। এখন কী করবে? ব্যাশ স্ক্রিনে প্রমাদ বার্তা বা এরর মেসেজ দেবে, '`cat: cdtemp: No such file or directory`'। এক একটা আদেশ পালনে সিস্টেমের এক এক ধরনের অপারগতার জন্যে এক এক রকম প্রমাদ বার্তা। প্রমাদ-বার্তাটা একটা ফাইলেও পাঠানো যায়, সে কথায় আসছি। এই কমান্ডটা দেখুন, '`cat cdtemp 2>errorfile`'। এর আগে অনস্তিত্ব ফাইল ক্যাট করার কমান্ড দেওয়ায় যে প্রমাদবার্তাটা ব্যাশ স্ক্রিনে দেখাচ্ছিল সেটাকে এবার চালান করে দিচ্ছে 'errorfile' নামের ডিস্কফাইলে। চালান করার চিহ্নটা, '>', আমরা চিনি, শুধু এখানে ওই '2' সংখ্যাটা আমরা চিনিনা, দুনম্বরিটা চিনে যাব, ক্রমে আমরা ব্যাশিক্ষিত হচ্ছিনা? 'errorfile' ফাইলটাকে যদি এবার স্ক্রিনে পড়ি, '`cat errorfile`' কমান্ড দিয়ে, ঠিক ওই এরর মেসেজটাই পাব, যেটা এর আগে ব্যাশ স্ক্রিনে দেখাচ্ছিল।

ব্যাশের এই প্রমাদ বার্তাই চিহ্নপ্রবাহের তিন নম্বর ফাইল, স্ট্যান্ডার্ড এরর। ছবিতে দেখুন, উপরে বাঁদিকে স্ট্যান্ডার্ড ইন, বা ডানদিকে স্ট্যান্ডার্ড আউট, দুটোই এখন চেনা। এবার এর সঙ্গে এসেছে স্ট্যান্ডার্ড এরর। স্ট্যান্ডার্ড এররের আবার দুটো গতি। হয় স্ক্রিনে পৌঁছনোর ডিফল্ট, নয়ত কোনো ডিস্কফাইলে রিডাইরেক্ট হওয়া। এইমাত্র যেমন করলাম, 'errorfile'-এ। ব্যাশ প্রমাদবার্তা পাঠিয়েছিল, কারণ '`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 রকম 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 মানেই সেই চিহ্নপ্রবাহটাকে রিডাইরেক্ট করা। যেখানে তার ডিফল্টে যাওয়ার কথা, সেখানে যেতে না-দিয়ে অন্য কোথাও নিয়ে যাচ্ছি, রিডাইরেক্ট করছি। এই যে '`2>errorfile`' অংশটা যোগ করলাম, এর মানে, এরর বার্তাটাকে তার ডিফল্ট গতি মানে স্ক্রিনে যেতে না-দিয়ে ঠেলে দিচ্ছি একটা ফাইলে, যার নাম 'errorfile'। তার আগে প্রমাদবার্তাটা স্ক্রিনে পাচ্ছিলাম। স্ট্যান্ডার্ড এরর বোঝাতে '2' ব্যবহার করতে হল, কিন্তু, '0' বা '1'-এর বেলায় সচরাচর হয়না, কারণ সেটাই ডিফল্ট। ধরুন, ডিরেক্টরির সমস্ত ফাইলের তালিকা বানাচ্ছি 'ls' কমান্ডের স্ট্যান্ডার্ড আউটকে রিডাইরেক্ট করে, '`ls > filelist`' দিয়ে। আদতে যা করছি

তা হল, 'ls 1> filelist'। '1'-টা আলাদা করে উল্লেখ করতে হয়নি, কারণ ওটা ডিফল্ট। কখনো কখনো করতে হয়, একাধিক প্রবাহকে একই ফাইলে রিডাইরেক্ট করতে হলে। কিন্তু সে ব্যাশের অনেকটা অগ্রবর্তী কমান্ডের ব্যবহারে। খুব যদি কৌতূহল হয়, খেজুরগাছে হাঁড়ি বাঁধো ম্যান, ব্যাশ পড়ার নেশা করো ম্যান।

১৩।। কমান্ডের মধ্যে কমান্ড

একটা কমান্ডের আউটপুট সরাসরি আর একটা কমান্ডের ইনপুট হচ্ছে, এটা আমাদের চেনা। বহুবার ব্যবহার করেছি। একে বলে পাইপ করা, যার চিহ্ন '|'। পাইপ ছাড়া আরো কয়েকভাবে একটা কমান্ডের মধ্যে আর একটা কমান্ড ব্যবহার করা যায়। তার একটার কথা আগে এসেছে, ব্যাককোট বা '`'। চিহ্নটা থাকে কিবোর্ডের একদম বাঁয়ে উপরে, ঠিক 'Esc' বা 'এসকেপ' চাবিটার নিচেই। 'writefiles' ব্যাশস্ক্রিপ্টে এই চিহ্নটা ব্যবহার করেছিলাম। স্ক্রিপ্টে একটা কাউন্টার ভ্যারিয়েবল ছিল, যার নাম 'c', যার ব্যাখ্যাটা নয় নম্বর দিনে ২.২ সেকশনে 'writefiles' স্ক্রিপ্টের লাইন ধরে ধরে আলোচনায় এসেছিল। লুপ প্রতিবার ঘোরার সঙ্গে সঙ্গে 'c' ভ্যারিয়েবলটার মান এক করে বাড়ানো হচ্ছিল। আজকের ৯ নম্বর সেকশনে 'writefiles' স্ক্রিপ্টে দেখুন, 'cat $i>> $f', মানে, প্রতিবার একটা করে নতুন ফাইল ক্যাট করছে, আর 'c'-এর মান এক করে বাড়িয়ে দিচ্ছে। বাড়াচ্ছে কারণ, প্রতিটি ফাইল ক্যাট করার আগে তার হেডিং-এ ফাইলের সিরিয়াল নম্বরটা লিখতে হচ্ছে ব্যাশকে। 'c=`expr $c + 1`' লাইনটা এই কাজটাই করছে। স্ক্রিপ্টের গোড়াতেই অ্যাসাইন করা আছে, 'c=1'। প্রথম ফাইলের হেডিং-এ আসবে এক, পরের ফাইলে আসবে দুই, এই ভাবে বাড়বে। কাউন্টার বস্তুটা প্রোগ্রাম লেখার খুব গোড়ার জিনিষ।

'expr' অঙ্ক কষার কমান্ড। ম্যানপেজের এক নম্বর সেকশনেই পাবেন। দুই ব্যাককোট চিহ্নের মধ্যের অংশটাকে ভাবুন। 'expr $c + 1'। এটা নিজে আলাদা একটা কমান্ড। '$c' মানে 'c' ভ্যারিয়েবলের মান। তার সঙ্গে '1' যোগ করে যোগফলটাকে দেয় 'expr $c + 1' এই গোটা কমান্ডটা। 'c=`expr $c + 1`' লাইনটা এই যোগফলটাকে অ্যাসাইন করে 'c' ভ্যারিয়েবলে। '=' চিহ্নটা ধার্য বা অ্যাসাইন করে, আগেই বলেছি। শুরুর লাইনে ব্যাশকে জানিয়েছেন, 'c=1'। মানে, 'c'-এর মান ধার্য করো '1'। এইমুহূর্তে ব্যাশ জানে 'c' হল '1', মেশিনের স্মৃতিতে 'c' ভ্যারিয়েবলের জমিতে লিখে নিয়েছে '1' সংখ্যাটা। এবার প্রথম ফাইল ক্যাট করা হল। ক্যাট করার পরেই এল ব্যাককোট লাগানো লাইনটা, 'c=`expr $c + 1`'। এখানে পেল 'expr $c + 1', মানে 'expr'-এর করা যোগফল। 'c' ভ্যারিয়েবলের মানের সঙ্গে '1' যোগ করে কী পাবে? তার কাছে আছে, 'c'-এর মান '1', এর সঙ্গে '1' যোগ করে যোগফল হবে '2'। এটাকে অ্যাসাইন করে দিল 'c'-তে। এখন 'c'-এর মান '2'।

ধরুন কমান্ড প্রম্পটে কাজ করছি। প্রথম কমান্ডে অ্যাসাইন করলাম, 'c=1', এন্টার মারলাম। দ্বিতীয় কমান্ড দিলাম, 'expr $c + 1'। ব্যাশ স্ক্রিনে ফুটিয়ে তুলল, '2'। 'expr' কমান্ডের অঙ্কটা ব্যাশ কষে নিল। এখন 'echo $c' দিলে ব্যাশ দেখাবে '1', কারণ 'c'-র ওই মানটাই ব্যাশকে দেওয়া আছে। এবার, তৃতীয় কমান্ড দিলাম, দ্বিতীয় কমান্ডের যোগফলটাকে 'c'-তে অ্যাসাইন করে, 'c=`expr $c + 1`'। এখন 'echo $c' দেখাবে '2'। অর্থাৎ, ব্যাককোট বা '`' চিহ্নের মধ্যের অংশটা, যার মান '2', তাকে ব্যাশ নিজেই হিশেব করে যোগফলটা অ্যাসাইন করে নিয়েছে 'c' ভ্যারিয়েবলের জমিতে। অ্যাসাইনমেন্ট কমান্ডের '=' চিহ্ন দেওয়া বিবৃতির মধ্যে আর একটা কমান্ড 'expr' ব্যবহার করা গেল ব্যাককোট চিহ্নের দৌলতে। স্ক্রিপ্টটা ভাবুন এবার। প্রতিবার, 'cat' করার পরেই ব্যাশ 'c' ভ্যারিয়েবলের সেই মুহূর্তের মান পড়ে নিচ্ছে মেমরি থেকে, সেই মানটার সঙ্গে 'expr' কমান্ড ব্যবহার করে '1' যোগ করে নিচ্ছে, যোগফলটাকে অ্যাসাইন করছে 'c' ভ্যারিয়েবলের জমিতে। প্রতিবার 'c' এক করে বেড়ে যাচ্ছে। প্রত্যেকটা ফাইলের হেডিং-এ আসা সংখ্যাটা আগের ফাইলের সংখ্যাটার থেকে এক বেড়ে যাচ্ছে। আমরা যেভাবে গুনি, সেই গোনার প্রক্রিয়াটাকে স্ক্রিপ্টিং ভাষা ব্যবহার করে ব্যাশের হাতে দিয়ে দিচ্ছি। নয়মান সাহেবের 'CC' বা কেন্দ্রীয় নিয়ন্ত্রণ মেশিনের ভিতরে পাঠিয়ে দিচ্ছি, ক্যালকুলেটরে যা করা যেতনা। কমান্ড সাবস্টিটিউশনের কাজটা আর এক ভাবে করা যেত '$' চিহ্ন দিয়ে। 'writefiles' স্ক্রিপ্টের 'c=`expr $c + 1`' লাইনটা মুছে, তার জায়গায় 'c=$(expr $c + 1)' লিখে দিলেও একই কাজ হত। আমরা আগেই জানি, '$' মানে মান। '$c' মানে 'c'-এর মান, তার সঙ্গে 'expr' দিয়ে '1' যোগ করছি। '(expr $c + 1)'-কে এবার দেখছি একটা ভ্যারিয়েবল হিশেবে, যার মান '$(expr $c + 1)'। সেই মানটাকে অ্যাসাইন করে দিচ্ছি 'c' ভ্যারিয়েবলের জমিতে।

কমান্ডের মধ্যে কমান্ড টেনে আনার আর একটা প্রক্রিয়া বেশ মজার। তার নাম 'tee', গাছেরটা খাওয়া যায়, এবং একই সঙ্গে, একটুও না-থমকিয়ে না-চমকিয়ে তলারটাও কুড়োনো যায়। আমাদের আজকের আলোচনার ২ নম্বর সেকশনে '.bashrc' ফাইলটার সঙ্গে মুখোমুখি হয়েছিলাম। কয়েকটা লাইন নিয়ে আলোচনাও হয়েছিল। শেষ চারটে লাইন তখন বাদ দিতে, না-ভাবতে বলেছিলাম। সেই লাইন চারটের সবগুলো বোঝার অবস্থা এখনো আসেনি আমাদের। শুধু, '~/.bashrc' ফাইলের লাইনগুলো থেকে মধ্যের একটা লাইন এবার পেড়ে ফেলার সময় এসেছে। লাইনটা ছিল, `'echo -e '\n***\n'>>F;/usr/bin/fortune -s | tee -a F | cat'`। লাইনটায় ব্যবহার হয়েছে মোট চারটে কমান্ড, 'echo', 'fortune', 'tee', এবং 'cat'। `'echo'` কমান্ডের পরে `'-e'` অপশনটার মানে আমরা জানি, '\n' দেখলেই একো একটা করে লাইন ছাড় দিয়ে পরের লাইনে চলে যায়। এই লাইনে 'echo'-কে আদেশ দেওয়া হয়েছে, '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 চিহ্ন '>' ব্যবহার না-করে, অ্যাপেন্ড বা যোগ করার '>>' চিহ্ন দিয়েছি। শুধু '>' চিহ্ন থাকলে, ডিরেক্টরিতে থাকা পুরোনো 'F' ফাইল মুছে নতুন একটা 'F' নামের ফাইল খুলে নিত ব্যাশ। কিন্তু '>>' চিহ্ন থাকায় তা না-করে, পুরোনো 'F' ফাইল থাকলে সেই ফাইলের শেষে যোগ করে দেবে কাঙ্খিত অংশটা, আর 'F' নামের কোনো ফাইল না-থাকলে, ওই নামে একটা ফাইল বানিয়ে, সেই ফাইলে লিখে দেবে। এই 'echo -e '\n***\n'>>F' অংশের পরে `';'` চিহ্ন। সেমিকোলন ';' চিহ্ন মানে ব্যাশের কাছে দুটো আদেশের মধ্যে যতি। এর পরে, '/usr/bin/fortune -s'। এটাই মূল 'fortune' কমান্ড যা কমান্ড প্রম্পটে দিলে একটা করে বাণী শোনায় ফরচুন। `'-s'` অপশনটা ফরচুনকে জানায়, একটু ছোট বাণী দিও বস, শর্টে মেরো। '/usr/bin' হল 'fortune' বাইনারির ঠিকানা। '/usr/bin/fortune -s' অংশের পরে পাইপ চিহ্ন '|'। অর্থাৎ, ফরচুন-বাণী সরাসরি স্ক্রিনে না-দেখে পাইপ করে দিচ্ছি। এটা চেনা ব্যাপার। এরকম কমান্ড চিনি, `'fortune -s >> F'`। ছোট সাইজের ফরচুন-বাণী স্ক্রিনে না-দেখে 'F' ফাইলে চালান করে দাও। এখন পর্যন্ত এটাই চিনি আমরা। হয় স্ক্রিনে পড়ো, নয় ফাইলে রাখো, দুটো এক সঙ্গে হয় না। কিন্তু, এই পাইপ চিহ্নের পরে 'tee' কমান্ডটার সেটাই মজা, কেকও খাবো, আবার তার ডো-টাও রয়ে যাবে। এই প্রথম পাইপের ডানদিকে আছে 'tee -a F | cat', যার মধ্যে একটা 'tee -a F' অংশ, তারপর আবার একটা পাইপ '|', তারপর 'cat'। `'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' চিহ্নপ্রবাহটাকে 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-বাণী যোগ হতে থাকল, স্ক্রিনে পড়তেও পেলাম। `man tee ...` ।

১৪।। অবস্থানগত নিয়ন্তা — এবং তাদের বদলানো

ব্যাশের বিল্ট-ইন ডিফল্ট সিস্টেম ভ্যারিয়েবলগুলো দেখার কাজে 'set' আমরা ব্যবহার করেছি। এটা 'set' কমান্ডের দৃষ্টিতে দেখলে, অত্যন্ত অপমানজনক। রোমনা হলিডে দিয়ে রোম চেনার মত। কিম্বা বলা, মাছি-নিয়ন্ত্রণে লিটলবয় এবং ফ্যাটম্যানের কোনো তুলনা হয়না, হিরোশিমা নাগাসাকিতে অন্তত বছরখানেক কোনো মাছি ছিলনা। ব্যাশের ভিতরের বিল্ট-ইন কাজগুলোর কথা এনেছি আগেও। অ্যালিয়াস বা এক্সপোর্ট ইত্যাদির সূত্রে। ব্যাশ ম্যানুয়ালে পাবেন 'set' নামের বিল্ট-ইন কমান্ড এবং অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটারদের কথা। কমান্ড প্রম্পটে 'set' মেরে কী হয়, সেটা দেখেছি, সমস্ত চালু ব্যাশ ভ্যারিয়েবল আর ফাংশনের তালিকা তৈরি করে দেয় স্ক্রিনে। এই 'set' দিয়েই একটা বিপুল কাজ করা যায়, নিজের প্রয়োজনের প্যারামিটার ব্যাশকে দিয়ে দেওয়ার কাজ। তার পর থেকে নিজের ইচ্ছেমত নিয়ন্তা বা প্যারামিটারগুলোকে নিজের খুশি মত ব্যবহার করা যায়। প্রথমে একটা মজার উদাহরণ দিয়ে বিষয়টা বুঝে নেওয়া যাক। কমান্ড দিন

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

এন্টার মারুন। কমান্ড প্রম্পট ফেরত এসেছে, মানে যা ঘটার ঘটে গেছে। এটা ব্যাশেষত্ব। ব্যাশের বিশেষত্ব। একজন আদ্যোপান্ত নীরব কর্মী। যে কাজ করতে দিয়েছেন, নিঃশব্দে করে দেবে। রব তৈরি করবে একমাত্র তখনই যখন কোথাও কোনো প্রমাদ ঘটবে। স্ট্যান্ডার্ড এররকে তখন তার ডিফল্টে মানে স্ক্রিনে শো করে দেবে। এখানে ব্যাশ কোনো উচ্চবাচ্য করল না, মানে, যা করার করা হয়ে গেছে। কিছু একটা ঘটে গেছে মেশিনের ইলেকট্রনিক উদরে। এখুনি মালুম পাওয়া যাবে। এবার কমান্ড দিন

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

হুবহু যা লেখা তা যদি টাইপ করে দিতে পারেন, নিজের আইডেন্টিটি ক্রাইসিস থেকে একদম হাতে হাতে একটা মুক্তি এবং সমাধান মিলে যাবে। আর আইডেন্টিটি ক্রাইসিস আমাদের থাকবেই, এটাই উপনিবেশ। সমাধান মিলল? খুব বেশি খচে যাওয়ার কোনো কারণ নেই। এমন নয় যে ফাঁকতালে আমি আপনাকে গাল দিয়ে নিলাম। ভাবুন, এটা এখানে লেখার আগে করে দেখে নিতে হয়েছে। তখন হুবহু ওই আত্মজ্ঞান আমাকেও পেতে হয়েছে। আর এত চটার কী আছে? 'রক্তবরণ-মুগ্ধকরণ-নদীপাশে-যাহা-বিঁধিলে-মরণ' নিজেই মুগ্ধ উত্তেজিত বিহ্বল হয়ে বলেছিলেন, ''উট উটঅ উঠঅ: সে তো মিস্টিরিয়াস অ্যানিমাল মশাই'' (কার্টসি জটায়ু)।

এবার দেখা যাক, এখানে কী ঘটছে। 'whoami' কমান্ডটার কথা আগে বলেছি কিনা মনে নেই। না, 'whoami' কোনো গভীর দার্শনিক আত্মজিজ্ঞাষা নয়, নিতান্তই চালু ব্যাশ প্রসেসের চালু ইউজারের নাম ফুটিয়ে তোলার কমান্ড। সেটা এখানে দুটো ব্যাককোটের মধ্যে। মানে, কমান্ড সাবস্টিটিউশন, আদেশ-বদল। কমান্ডের মধ্যে কমান্ড, 'whoami' কমান্ডের ফলাফল 'echo' কমান্ডের মধ্যে টেনে আনছি। পরের অংশটা, '$3 $4 $8' — এটা 'set' কমান্ডের অবদান। ঠিক আগেই যে লাইনটা 'set' কমান্ডে দিলাম, সেখানে 'set' শব্দটার পর থেকে গুনুন, ১-নম্বর শব্দ 'Arab', ২-নম্বর 'kafela'। এরকম করে ৩, ৪, আর ৮ নম্বর শব্দ তিনটে কী দেখে নিন। আত্মজ্ঞানের রহস্যভেদ হল? 'set' ঠিক এটাই করে। যে শব্দগুলো আসছে সেগুলোকে স্পেসে স্পেসে আলাদা করে এক দুই তিন চার — পরপর সংখ্যায় অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটার হিশেবে তুলে রাখে। এবার যখনই দরকার পড়ে, আমরা তাদের '$1', '$2', '$3' ইত্যাদি নামে ডাকি, এবং বৈদ্যুতিন অন্ধকার থেকে ডাকিনীমন্ত্রে তাদের জাগিয়ে তোলে ব্যাশ। এবার ব্যাশকে দিয়ে একটু কম অর্থহীন একটা কাজ করানো যাক। আর এতে '~/.bashrc' ফাইল বদলানোর অভ্যেশটাও একটু 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, কমান্ড দিন '`source ~/.bashrc`', এতে লগ-আউট লগ-ইন না করেই বদলে যাওয়া '~/.bashrc' ফাইলের বদলানো কনফিগারেশন ব্যাশের মধ্যে নিয়ে আসা যায়। অথবা, জ্যান্ত রকমে মানে ইন্টারাক্টিভলি একটা ব্যাশ শেল চালু করুন, ব্যাশের ভাষায় ইনভোক। মানে জাস্ট কমান্ড প্রম্পটে '`bash`' লিখে এন্টার মারুন। এটা আমি, মানে ইউজার 'dd' করায় ব্যাশ দেখাল

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

'`$()`'-এর জায়গায় '``' চিহ্ন দিয়ে কমান্ড সাবস্টিটিউট করেও করা যেত '~/.bashrc'-র এই তিন লাইন বদল। প্রথম লাইন দুটো লেখাই যেত, '`set `date``' এবং '`echo Hello Dear `whoami``'। তাতেও একই ফল পেতাম ব্যাশ প্রম্পটে। প্রথম লাইনের 'date' কমান্ডটা সাবস্টিটিউট করা হচ্ছে 'set' কমান্ডের মধ্যে। এটা কাজে লাগছে তিন নম্বর লাইনে, '`echo Today is $1-day, $2, $6, and the time is: $4`'। এই লাইনটা বোঝার চেষ্টা করার আগে একবার 'date' কমান্ডটার কাজটা একটু বুঝে নিন। প্রম্পটে আমি '`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 দেখুন, লগ-ইন করার মুহূর্তে ব্যাশ যা পড়েছিল তার থেকে আমার 'date' কমান্ডটা দেওয়ার মধ্যে 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।

এবার, আমাদের সাবস্টিটিউট করা 'date' কমান্ডকে 'set' দিয়ে ব্যাশ কী ভাবে অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটার বানিয়ে নিল, সেটা খেয়াল করুন। ব্যাশ নিজে 'date' কমান্ডটা দিয়ে যা পেয়েছিল, তার পুরোটা 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। শুধু 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, 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

ব্যাশ বা শেল আমার নিজের বেশ উত্তেজক লাগে। উত্তেজনাটা চারিয়ে গেছিল স্টিফেন প্রাটার 'অ্যাডভান্সড ইউনিক্স এ প্রোগ্রামারস গাইড' বইটা থেকে। সত্যি বলছি, কলেজ থেকে ফেরার সময় নটা-পাঁচের বনগা লোকালে অন্যের-কান-চুলকে-দেওয়া ভিড়ে দাঁড়িয়েও পড়ে দেখেছি, বেশ পড়া যায়। ট্যানেনবম-এর খুব নিকট প্রতিদ্বন্দ্বী। আমার কম্পিউটার ভাবনার যেটুকু যতটুকু, প্রায় আগাগোড়াই এই দুটো বইয়ের কল্যাণে তৈরি। এখানে একটা ব্যথা আছে। এ দুটো বইয়ের একটাও গ্নু-লিনাক্সের জন্যে কাস্টম-বিল্ট নয়। ট্যানেনবমে তবু গ্নু-লিনাক্সের কিছু উল্লেখ আছে। আর প্রাটার বইটার ভারতীয় সংস্করণই ছিয়াশির, মানে, লিনাস তখনো তার কারনেলই নামায়নি নেটে। প্রাটার এই বইটা বর্ন শেল নিয়ে লেখা। বর্ন-এগেন শেল বা 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। নিজেরাই বাংলায় গ্নু-লিনাক্সের বই লিখুন না, যে যে পারেন, অন্যের সঙ্গে নিজেও লাভ পাবেন। পাঠমালাটা লিখতে লিখতে আমি নিজেই অনেক স্পষ্ট ভাবে শিখে গেছি গোটাটা। কোনো কিছু শেখার জন্যে লেখার মত ভালো কোনো উপায় আর হয়না। আর এভাবেই, সবাইকে মিলিয়ে, 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 হাঁটু বেঁকে যাচ্ছে, মুখ কুঁকড়ে যাচ্ছে, খোয়া পাথরে এতটাই লাগছে, কিন্তু তাতে দায়িত্বপালনের কোনো খামতি ঘটছে না। পোস্টমডার্ন তত্ত্বে মানবজাতির ইতিহাসকে ব্যক্তিমানবের অতিক্রম করে যাওয়ার গল্প বারবার লিখি, লিখি যে, সেটাই প্রতিরোধ। তার এর চেয়ে বড় উদাহরণ আর হয় না। চারদিকে যখন এ ওরটা কেড়ে খাওয়া, অন্যের খিদের থেকে চোখ ফিরিয়ে রাখাটাই নিয়তি ভবিতব্য এবং সামাজিক কানুন, তখনো তো প্যাঁচা জাগে, পাথরের পথে বাচ্চাতর-কোলে বাচ্চা দাঁতমুখ খিঁচিয়ে পথ হাঁটে। তাহলে আমরা পারিনা কেন? নিজেদের মতো করে নিজেদের রেজিস্ট্যান্সের ব্যাকরণ আসুন নিজেরাই বানাই। কত ভাবে বানানো যায়। কোনো বাঁধাধরা নিয়ম নেই। আসুন, গ্নু-লিনাক্স দিয়েও আমরা খিদে জিইয়ে রাখার রাজনীতির বিরুদ্ধে এক পিস প্রতিরোধ বানাই। কাঠবিড়ালি তার পিঠে রামের পাঁচ-আঙুলের কৃতজ্ঞ সমাদরের দাগ নিয়ে গাছের ডালে ডালে লেজ বেঁকিয়ে লাফিয়ে বেড়ায়, আমাদের তো রামটুকুও নেই, না-থাকুক, সেতুবন্ধের কাজে নিজের নিজের ব্যক্তিগত বাদামের খোলা থাকবেনা কেন?

ব্যাশের নিজেরই বরাদ্দ করা বিশেষ কিছু ভ্যারিয়েবল নিয়ে বলছিলাম। 'PATH', 'PS1' ইত্যাদি এনভায়রনমেন্ট ভ্যারিয়েবলকে ধারণ করার ব্যাশপ্রক্রিয়ার কথা আগেই বলেছি। এর সঙ্গে ব্যক্তিগত লোকাল ভ্যারিয়েবল বানানো এবং ব্যবহারের কায়দাও চিনেছি। কিছু বিশেষ ব্যাশ ভ্যারিয়েবলকে এবার দেখব আমরা, প্রাটা এদের স্পেশাল শেল ভ্যারিয়েবল বলে ডেকেছেন। এদের আর আলাদা করে বানিয়ে নিতে হয়না, বানানোই থাকে। অবস্থানগত নিয়ন্তা বা পোজিশনাল প্যারামিটারদের পরেই এদের আসার একটা কারণ আছে, এখুনি দেখতে পাবেন। অঙ্কের সঙ্গে যাদের

পরিচয়টা একটু কম, তাদের জন্যে একটু ভ্যারিয়েবল আর প্যারামিটার ধারণাদুটো নাড়িয়ে নিই। ধরুন একটা ছাত্রের রেজাল্ট নিয়ে ভাবছেন, তাকে বলছেন, পড়ার সময় বাড়া, সিনেমা দেখিস না, আড্ডা কম দে, ইত্যাদি। কিন্তু একবারো বলছেন না, তোর বাবার আয়টা একটু বাড়া, তোর প্রতি বাবামার ভালোবাসাটা বাড়া, তোর চারপাশে বায়ুমণ্ডলের দূষণটা কমা যাতে হাঁফানিটা কমে, ইত্যাদি। কিন্তু ভেবে দেখুন, এর সবগুলোই সমান জরুরি। বলেন না, কারণ, এই মুহূর্তে ছাত্রটা বা আপনি, আপনারা কেউই চেয়ে চেয়ে মরে গেলেও, বা তার চেয়েও সাংঘাতিক, বিপ্লবী হয়ে গেলেও, কিছু জিনিষ কিছুতেই বদলাতে পারবেন না। সীমাবদ্ধ অধিকারের ভূমিতে, ছাত্রটার লেখাপড়ার উন্নতির জন্যে মাত্র কয়েকটা জিনিষই বদলাতে পারেন। তারাই ভ্যারিয়েবল। আর, যারা অপরিবর্তনীয় থাকছে, স্থির রয়ে যাচ্ছে, তারাই প্যারামিটার। অবশ্য পরিস্থিতি থেকে পরিস্থিতিতে অবস্থাটা বদলায়, প্যারামিটাররাও ভ্যারিয়েবল হয়ে ওঠে। অর্থাৎ, একটা বিশেষ পরিস্থিতিতে যা বদলায়-না তা প্যারামিটার, আর, যা বদলায় তা 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 নামটাই চলে দেখেছি। চলে যখন, তার কোনো একটা আলাদা অর্থ বা ইতিহাস আছে নিশ্চয়ই, যা 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 প্যারামিটারগুলো দেওয়ার সময়, সেই আদেশে প্যারামিটারের মোট সংখ্যা। 'উট' ডাকার উদাহরণটায়, 'set' করার পর, ব্যাশ প্রম্পটে '`echo $#`' কমান্ড দিলে ব্যাশ দেখাত, '`12`'। গুনে দেখুন, বারোটাই শব্দ আছে। আবার 'set' কমান্ড দেওয়ার বা লগ-আউট করার আগে অব্দি এটা লাগু থাকবে।

$* বা $@ — 'set' দিয়ে মোট যত প্যারামিটার দেওয়া হয়েছে, তার পুরো তালিকাটা দেয় এরা দুটোই। দুয়ের মধ্যে একটাই তফাত ঘটে যখন কোট চিহ্নের মধ্যে দেওয়া হয়। লগ-ইন করার সময়, '~/.bashrc'-তে '`set $(date)`' কমান্ডের পর, '`echo $*`' বা '`echo $@`' করলে, লগ-ইনের মুহূর্তে 'date' কমান্ড যা দিয়েছিল, সেই গোটাটা ব্যাশ দেখিয়ে দেবে। বা উটের গল্পে, আরব কাফেলার গোটা বাক্যটা। প্রতিবার 'set' দেওয়া মাত্র আগেরটা মুছে নতুন মান ভরে দেওয়া হয় পোজিশনাল প্যারামিটারের এই ভূমিটায়।

$! — '`echo $!`' কমান্ডে ব্যাশ শেষ ব্যাকগ্রাউন্ডে পাঠানো প্রসেসটার প্রক্রিয়া-সূচক বা পিআইডি (PID) দেখায়। ব্যাকগ্রাউন্ডের আলোচনা আগেই করেছি। অ্যাম্পারস্যান্ড (`&`) চিহ্নটা আমরা চিনি, একটা প্রসেসকে সক্রিয় সম্মুখভূমি থেকে আপাতনিষ্ক্রিয় পশ্চাৎভূমিতে ঠেলে দেয়। '`bash &`' কমান্ড দিয়ে ব্যাকগ্রাউন্ডে একটা ইন্টারাক্টিভ শেল প্রসেস চালু করলেই ব্যাশ দেখায়, '`[1] 1301`' জাতীয় কিছু একটা। ব্রাকেটের মধ্যে '`1`' হল জব নাম্বার বা বকেয়া কাজের সূচক, আর '`1301`' এর পিআইডি। '`echo $!`' কমান্ড দিলেও এখন এই পিআইডিটা দেখাবে, '`1301`'। আগেই করেছি এগুলো, মনে পড়ছে?

$$ — '`echo $$`' দেখায় চালু ব্যাশ প্রসেসের পিআইডি, যার মধ্যে দাঁড়িয়ে কমান্ডটা দিলেন। আগের প্যারায় ব্যাকগ্রাউন্ডে পাঠানো ব্যাশ প্রসেসকে '`fg`' মেরে ফোরগ্রাউন্ডে আনুন। ব্যাশ চালু হয়ে গেল। এখন চালু ব্যাশটা হল ব্যাশের পেটে ব্যাশ। এবার কমান্ড দিন '`echo $$`'। ব্যাশ দেখাবে, '`1301`'। আগের প্যারাতেই এই পিআইডিটা পেয়েছিলাম, ব্যাকগ্রাউন্ডে পাঠানোর পরে, যাকে এখন ফোরগ্রাউন্ডে আনলাম। মিলে গেল।

$0 — এই মুহূর্তে ব্যাশ যে কমান্ডটা পালন করছে, তার নাম ভরা থাকে '$0'-তে। একটু আগের 'writefiles' স্ক্রিপ্টটার একদম শেষ লাইনে, মানে, '`done`'-এর পরের লাইনে যোগ করে একটা নতুন লাইন যোগ করে দিন, '`echo -e "\n\n$d\n[Made by $0 on $(date)]">>$f`'। এবার 'writefiles' চালিয়ে যেমন বানিয়েছি, তেমনি একটা ফাইল বানান। সেই শেষে নিচের অংশটা যোগ হয়ে যাবে। কেন?

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

$? — শেষ ব্যাবহৃত ব্যাশ কমান্ডের এক্সিট স্ট্যাটাস বা নিষ্ক্রমণ স্থিতি দেখায়। মানে, কমান্ডটা তার কাজ করতে পেরেছে কি পারেনি। এই নিরিখে ব্যাশ বেশ নাগার্জুনবাদী, তার কাছে সাফল্য হল শূন্য। যদি '`echo $?`' করলে '`0`' দেখায়, তার মানে কাজটা হয়েছে। প্রোগ্রামিং ভাষা সি-তেও কিন্তু তাই। এটাই বা কেন?

$- — ব্যাশ শেলের অপশান। 'echo $-' কমান্ডটা দিলে স্ক্রিনে আমরা দেখতে পাই, যে ব্যাশ প্রসেসের মধ্যে দাঁড়িয়ে আছি, সেটা কী কী অপশান নিয়ে চালানো হয়েছে। ম্যান পড়ে দেখুন, গাদা গাদা অপশান আছে। নানা ভাবে চালানো যায় ব্যাশ। আর এর আগের দুটো প্যারায় দুটো না-উত্তর-দেওয়া প্রশ্ন আছে। ওগুলো টাস্ক।

১৬।। মেটাক্যারেকটার, গ্লব এক্সপ্রেশন

অতিচিহ্ন বা মেটাক্যারেকটার জাতীয় রাশভারি নাম দিয়ে যাদের উল্লেখ করছি, সেই ভ্যাবলাকান্ত চিহ্নগুলোকে আমরা ব্যবহার করে আসছি পাঠমালার গোড়া থেকেই, শুধু নাম করিনি আলাদা করে। '>', '|', '\', '&', '&&', '*', '.' ইত্যাদি। এবার এদের একটু গুছিয়ে চিনব। এদের মধ্যে একটা গুরুত্বপূর্ণ অংশ হল পাইকিরি ছক বা গ্লব এক্সপ্রেশন। একজন একজন করে চুন-চুনকে মারুঙ্গার গল্পে যখন আর বিশ্বাস থাকেনা, গণহত্যাপন্থী হয়ে ওঠার সময় আসে, তারও ব্যাবস্থা করা আছে গ্নু-লিনাক্স সিস্টেমের ব্যাশ শেলে। আর একটা নামও আছে এদের, রেগেক্সপ বা রেগুলার এক্সপ্রেশন (Regular-Expression/REGEXP)। রংরুট খুচরো ইউজারের কাঁপা কাঁপা ভিতু হাতে একটা একটা করে ফাইল খোঁজা নয়, এক লপ্তে লাটকে লাট ফাইল পাইকিরি রকমে নাড়াচাড়া করার জন্যে লাগে এই পাইকিরি ছক বা গ্লব এক্সপ্রেশন। এবার তাকে বুঝব আমরা।

একটা অস্থায়ী এলেবেলে ডিরেক্টরি বানান, তাতে ফাইল বানান নিচের তালিকামত। সব ফাইলই মন্ত্রী-উবাচের মত, ফাঁপা এবং ফাঁকা। ইউজার টাচউড, টাচ করলেই ফাইল গজায়। নামগুলো এরকম বিকট ইচ্ছে করেই। পাইকিরি ছকের ব্যাকরণটা যাতে মালুম হয়। বিশ্বাস করুন, আমি আমার ফাইলের এত খারাপ নাম দিই-না, এত খারাপ ফাইলের নাম দেওয়া যায়না। এখানে এরকম নাম দেওয়ার কারণগুলো এখনি 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। ষোলটা প্রবন্ধ। ইংরিজির তিন, ভূগোলের চার, ইতিহাসের তিন আর অঙ্কের ছয়। ইংরিজি, ভূগোল, ইতিহাস আর অঙ্ক, চারটে বিষয়ের চারটে বিবলিওগ্রাফি এবং চারটে নোটস। শুধু ছোট হাতের নাম দিয়েছি, বড় হাতের অক্ষর ব্যবহার করিনি, অশালীনতাটা 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

এরা আছে হোম ডিরেক্টরির 'essay' সাবডিরেক্টরিতে। '~/essay' ডিরেক্টরিতে বসে, 'ls' মারলে, এই যাবতীয় ফাইল দেখতে পাব। 'ls *' মারলেও তাই হবে, আমরা জানি, '*' চিহ্নটার মানে সমস্ত কিছু, শুধু ডটানন ফাইলদের ধরতে পারেনা '*' চিহ্নটা। যদি কমান্ড দিই, 'ls essay.[gh]*.text', তাহলে আর সমস্ত ফাইল দেখতে পাব না, শুধু দেখব ভূগোল আর ইতিহাস, 'geo' আর 'hist' প্রবন্ধের তালিকা। ভূগোলের চার আর ইতিহাসের তিন। এই 'essay.[gh]*.text' 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	

'essay.[gh]*.text' ছকটাকে ব্যাশ কী ভাবে দেখছে সেটা বোঝা যাক। প্রথমে মেলাচ্ছে সেই সমস্ত ফাইল যাদের শুরুতেই আছে পাইকিরি ছকের গোড়ার অংশটা, 'essay.'। বিন্দুটাকে দেখতে ভুল করবেন না, বিন্দুতে যে কী বীভৎস সব কেলো ঘটে যায় তা ইউক্লিডের শ্বশুরও তেমন জানতেন না, গ্নু-লিনাক্স করতে গিয়ে আপনি যেমন জানবেন। তার মানে, 'essay.' মিলিয়ে ব্যাশ তুলে নিচ্ছে কুড়িটা ফাইল, এবং বাদ দিচ্ছে চারটে, কারণ তারা 'essay.' দিয়ে শুরু নয়। এবার ব্যাশ যাচ্ছে ছকের দ্বিতীয় অংশটায়, '[gh]', দেখছে চৌকো ব্রাকেটের ভিতরকার অক্ষরগুলো। এই কুড়িটা

ফাইলের মধ্যে যারা 'g' বা 'h' দিয়ে শুরু সেই সবাইকে সে রেখে দিচ্ছে হোলসেল ঝোলায়। অন্যদের বাদ দিয়ে দিচ্ছে। এই ধাপে বাদ পড়ে যাচ্ছে ইংরিজির মোট চারটে, তিনটে প্রবন্ধ আর একটা বিবলিওগ্রাফি, 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। এর মানে, এতে '.text' অংশটা থাকতেই হবে, তার আগে যাই থাকুক। মানে, সেই সমস্ত ফাইল যারা '.text' দিয়ে শেষ হয়েছে। এই ধাপে দেখুন পড়ে থাকা নটা ফাইলের বাদ পড়ছে দুটো ফাইল, 'essay.geography.biblio' এবং 'essay.history.biblio'। কারণ, এদের শেষে পাইকিরি ছকের শেষ অংশটা, '.text', নেই। নটা থেকে দুটো বাদ, পড়ে রইল ওই সাতটা ফাইল, আগের টেবিলে যাদের 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 '^' চিহ্নটার মানে, সেই অক্ষরেরা যারা 'g' বা 'h' নয়। ফাইলের তালিকাটা দিয়ে দিচ্ছি, বুঝে নিন, আলোচনার সঙ্গে মিলিয়ে। তফাত একটাই ঘটেছে, 'g' বা 'h' এই সমাহারটার বদলে সেই সমাহার যা তৈরি তাদের নিয়ে যারা 'g'-নয় বা '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

দুটো অক্ষর 'g' বা 'h' বোঝালাম '[gh]' দিয়ে, মাত্র দুটো অক্ষর বলে। যদি অনেকগুলো অক্ষরগুলো হত, তাহলে সেটাকে একটা ধারাবাহিকতা হিশেবেও দিতে পারতাম। ধরুন, শুধু ইংরিজির ভূগোলের আর ইতিহাসের দশটা প্রবন্ধই দেখতে চাইছি, তখন চাইব শুধু '`essay.[e-h]*.text`' ফাইলগুলোকে। '[e-h]' বোঝাচ্ছে যে, মধ্যের অংশটা 'e' থেকে 'h' অব্দি যে কোনো বর্ণ দিয়েই শুরু হতে পারে। এটা অন্য আর এক পাইকিরি ছকেও করা যেত, '`essay.{english,geo,hist}.text`'। সেটাও ধরবে সেই সমস্ত প্রবন্ধের ফাইলকে যাদের মধ্যের অংশটায় আছে, 'english' বা 'geo' বা 'hist'। এই সংক্রান্ত আরো প্রচুর আছে, ম্যান পড়ার নেশা করো ম্যান ... ।

শুধু অঙ্কের প্রবন্ধের ফাইল কটাকে যদি চাই, 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, জনগনতান্ত্রিক অঙ্কবিপ্লবের ভ্যানগার্ডদের দলে আছেন, এবং অঙ্কের প্রবন্ধ লিখেই চলেছেন, 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।

কমান্ড লাইনে পাইকিরি ছক ব্যবহারের একটা সমস্যা আছে। ধরুন, একটা অঙ্কের প্রবন্ধের নাম বদলাতে চান। কমান্ড দিলেন, '`mv essay.math.1.text math.1`'। ব্যাশ বিনা আপত্তিতে আপনার অঙ্কের প্রবন্ধটার সামনে আর পিছন থেকে 'essay' আর 'text' অংশদুটো উড়িয়ে দিল। কিন্তু এই কাজটা লাটে পাইকিরি ছকে করতে চাইলে, '`mv essay.math.*.text math.*`' কমান্ড দিয়ে দেখুন, ব্যাশ অস্বীকার করবে, প্রমাদ বার্তা পাঠাবে স্ক্রিনে। কিন্তু দিব্য এদের লাটে কপি করা যাবে। ডিরেক্টরি বানান, '`mkdir ~/math.bkp`' কমান্ডে। সব কটা অঙ্কের প্রবন্ধ কপি করে দিন '~/math.bkp' ডিরেক্টরিতে, '`cp essay.math.*.text ~/math.bkp/`', কমান্ড দিয়ে। সঙ্গে সঙ্গে হয়ে গেল, কোনো আপত্তি না-করে। '`ls ~/math.bkp/`' মেরে মিলিয়েও নিলেন ফাইলগুলো। কেন একটা করছে, অন্যটা করছে না? আসলে 'mv' কমান্ডের আভ্যন্তরীন কাঠামোটা খেয়াল করুন। কমান্ডটার দুটো অংশ। 'mv <name1> <name2>' হল তার ছকটা। 'mv' একটা ফাইলকে নড়ায় অন্য একটা

ফাইলে বা ডিরেক্টরিতে। যাকে নড়ায় তার নাম 'name1', নড়িয়ে যেখানে রাখে তার নাম 'name2'। 'name2' একটা ডিরেক্টরির নাম মানে, নড়িয়ে একটা ডিরেক্টরিতে রাখছে, ভৌত অর্থে নড়াচ্ছে। 'name2' একটা ফাইলনাম মানে, নড়িয়ে অন্য নামের ফাইলে রাখা হচ্ছে, ফাইলটার নাম বদলাচ্ছে 'mv'। 'mv' অন্য কোনো ছকে আদেশ নেয়না। এইমাত্র 'mv' আদেশটা এই ছক মেনেই দেওয়া, কিন্তু ঝামেলাটা পাকাচ্ছে ব্যাশ। পাইকিরি ছককে ব্যাশ বাড়িয়ে নেয়, এক্সপ্যান্ড করে নেয়। 'mv' কমান্ডের প্রথম অংশের পাইকিরি ছকটা বেড়ে গিয়ে হয়ে দাঁড়াচ্ছে ছ-টা ফাইল, মিলিয়ে দেখুন। 'mv' নিজের ছকটা খুঁজে পাচ্ছেনা, কাজও করছেনা। 'cp'-র কোনো সমস্যা নেই, ছ-টা ফাইলের পরে আসছে তাদের পাঠানোর ঠিকানা, কাজ করতেও কোনো অসুবিধে হচ্ছে না। ফাইলের নাম কি তাহলে লাটে বদলানো যাবেনা? প্রথমে ফাইলের বড় নামকে ছোট করে দেখাব, একটু মনোযোগ লাগবে ছোট করাটায়, তারপরে ছোট নামকে বড় করব। একটা ভ্যারিয়েবল বানানো যাক, 'filename' নামে, '`filename=essay.math.text`' কমান্ড দিয়ে। একবার '`echo $filename`' কমান্ড দিয়ে মিলিয়ে নেওয়া যায়, ঠিক দেখাচ্ছে কিনা। ব্যাশ ভুল করেনা, তবু দেখে নিতে দোষ কী? আমাদের 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 ${filename##essay.}
```

দেখুন, কমান্ড প্রম্পটে ফুটে উঠেছে, '`math.text`'। 'filename'-এর মুণ্ডটা বলি গেছে, গোড়ার '`essay.`' অংশটা। কিন্তু কেউ যদি লেজ বলি দিতে চায়, '`.text`' অংশটা ওড়াতে চায়? অন্য একটা কমান্ড দিন

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

এবার দেখুন, কমান্ড প্রম্পটে ফুটে উঠেছে, '`essay.math`'। বলির মন্ত্রটা বুঝে নিন এবার। 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 এতদিনে মসৃণ। এই মন্ত্রটা এবার লাটে প্রয়োগ করা যাক, পাইকিরি ছকে। ডিরেক্টরি তৈরি আছে। অঙ্কের প্রবন্ধ কপি করে '~/math.bkp' বানানো কেন, যদি-না বলির কাজে লাগে? '~/math.bkp' ডিরেক্টরিতে ঢুকে '`ls`' মারলে ব্যাশ যাবতীয় ফাইলকে দেখাবে।

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

প্রতিটি ফাইলের গোড়ায় একটা 'essay.' মুণ্ড, শেষে একটা '.text' লেজ। প্রথমে মুণ্ডবলি, পরে লেজ। কমান্ড দিন

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

এটা একটা 'for' লুপ, ধরে বোঝানো আছে আগে। 'ls *math*' করে, মধ্যে 'math' অংশটা আছে এমন সমস্ত ফাইল তোলা হচ্ছে, তাদের তালিকাটা কমান্ডের মধ্যে কমান্ডের ব্যাককোট দিয়ে নিয়ে আসা হচ্ছে। এবার তার থেকে ব্যাশ একটা করে ফাইল পড়ছে এবং তার নামের গোড়ার 'essay.' অংশটা ছেঁটে দিচ্ছে। আবার '`ls`' মারুন।

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

মানে, মুণ্ডবলি খতম। এবার লেজ। কমান্ড দিন

```
for i in `ls *math*`; do mv $i ${i%%.text.};done
```

কমান্ড প্রম্পট ফেরত এলে আবার '`ls`'। দেখুন, লেজহীন কবন্ধ ফাইলে ভরে গেছে '~/math.bkp'।

```
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, এক কমান্ডেই মুণ্ড আর লেজ গজাবে।

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

'ls' মারলেই দেখবেন, ফাইলগুলো পৌঁছে গেছে একদম প্রথম চেহারায়। 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 দিই, স্ক্রিপ্ট বানাতে গিয়ে বেশ কাজে লাগে। পথ-সহ একটা ফাইলের গোটা ঠিকানা থেকে নিছক ফাইলনাম আর পথ এই দুটোকে আলাদা করে বার করে নিতে পারে ব্যাশ, 'dirname' আর 'basename' কমান্ড দিয়ে। ধরুন, ঠিকানাসহ একটা ফাইল, '/usr/local/mplayer/conf'। এর মধ্যে প্রথম অংশটা, '/usr/local/mplayer', হল ফাইলটার ঠিকানা বা পথ মানে 'dirname'। এবং 'conf' হল ফাইলটার নিছক নামটুকু বা '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 তারা অনেক পরিণত স্ক্রিপ্ট, সহজ উদাহরণ মাথায় আসছেনা। খুব দরকারও নেই, এই অর্থে যে, আজকের এই আলোচনাটা স্ক্রিপ্ট শেখাতে চায়না, এগিয়ে দিতে চায় স্ক্রিপ্ট লেখার স্ক্রিপ্ট পড়ার দিকে। গ্নু-লিনাক্স সিস্টেমের মধ্যেই অজস্র স্ক্রিপ্ট আছে, তারা কোথায় কখন কী করে সেটা বোঝার জন্যেই দরকার স্ক্রিপ্ট পড়া। এই স্ক্রিপ্টগুলো তাদের লেখা যারা সবচেয়ে ভালো করে কাজটা জানে। শেখার জন্যে আদর্শ।

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)। 'set' কমান্ড দিয়ে ব্যাশের এনভায়রনমেন্ট ভ্যারিয়েবলদের পড়তে জানি আমরা। এবার সেই 'set' কমান্ডের ফলাফলকে একটু 'grep' গ্রেপ মারা যাক। 'set| grep 'IFS''। এতে যে ফলাফল আপনার স্ক্রিনে দেখাবেন ব্যাশবাবু, তাতে এজাতীয় একটা লাইন পাবেনই, 'IFS=$' \t\n''। লাইনের ডানদিকটা খেয়াল করুন। দুটো কোটের মধ্যে রয়েছে তিনটে জিনিষ। একটা ফাঁকা ভূমি, তারপর '\t' বা ট্যাব-চিহ্ন, তারপর নিউলাইন '\n'। মানে, এই তিনটের যে কোনোটাই ফিল্ড সেপারেটর বলে ধরে নেয় ব্যাশ। একটু আগে আমরা যখন 'set' কমান্ড দিয়ে পোজিশনাল প্যারামিটার বা 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 ক্যারেজ রিটার্ন নয়, ক্যারেজ রিটার্ন পাই এন্টার চাবি টিপে। মানে, এই আমার কমান্ড খতম, এবার বাছা ব্যাশ তুমি পিন্ডি চটকাতে শুরু করো। 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 চলে যায়, কিন্তু সেটা আসলে স্ক্রিনের নতুন লাইন, 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 হতে পারে একটা স্পেসের বা ট্যাবমার্কের বা নিউলাইনের।

সারিবদ্ধ আদেশের সেমিকোলন ';'

ইনিও আমাদের চেনা। আমার গদ্য লেখার শুরু যার হাতে, দান্তে, ভাস্করজ্যোতি সমাস করেছিল, দাঁত-অন্তে-যাহার — বহুব্রীহি, ছড়া চলত, দান্তের গাট্টা টাকায় আটটা, সেই 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। নয় নম্বর দিনে ‘/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 ডিডি।

```
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 (০) 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 'grep' কমান্ড সেই লাইনগুলোকে নিংড়ে আনতে পারে যাতে এই 'integer' শব্দটা আছে এবং স্ক্রিনে দেখিয়ে দিতে পারে। 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, 'grep' যদি পেয়ে যায় 'integer' শব্দটা তাহলে তো লাইনগুলোই দেখাবে, আর যদি না-পায় তখন দেখাবে তার বার্তা, 'NO 'integer' found'। অতিচিহ্ন মোটামুটি নামল। কিন্তু এদের নিয়ে এখনো একটা কুচো সমস্যা আছে। আগেই বললাম, এই অতিচিহ্ন বা মেটাক্যারেকটারগুলো 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 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 লেখার আগে নিজে একবার কমান্ড দিলাম, 'echo An asterix is a *.'। আমার বোধভাষ্যি মত, এখানে তো ফাইল তালিকা দেখানোর কথা, ব্যাশ তাই করে বলেই জানি। কিন্তু তা না-করে, দেখি 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 আলাদা করে নিয়েছি আগেই, আদেশ বদল বা কমান্ড সাবস্টিটিউশনের চিহ্ন হিশেবে। এই ব্যাকস্ল্যাশ, এককোট আর জোড়াকোট — '\', ''', '"' — এই তিনটে অতিচিহ্ন ব্যবহার 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। '\' চিহ্ন হলে টাইপ করতে হবে, '*\?*'। আর ''' চিহ্ন হলে টাইপ করতে হবে, ''*?*'''। আর '"' চিহ্ন হলে টাইপ করতে হবে, '"*?*"'। এই তিনটে কেসের প্রত্যেকটাতেই আপনি ফল পাবেন এক, ব্যাশ ফুটিয়ে তুলবে '*?*'। তাহলে এদের তফাতটা কোথায়? তফাত মালুম করার জন্যে আমাদের পুরোনো চেনা তক্তা '$PATH' আছে। তাতে একবার করে এই তিনটে পেরেকই মেরে দেখুন। তিনবার এই তিনটে কমান্ড দিন। 'echo \$PATH', 'echo '$PATH'' এবং '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 মেশিনের সুজে সিস্টেমে 'dd' নামের ব্যবহারকারীর ব্যাশের পথনির্দেশ,

```
/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
```

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। 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 '`'।

১৯।। নিয়ন্ত্রণ কাঠামো

পাঠমালার শেষ দিনের শেষ সেকশনে আবার ফিরে এলাম নিয়ন্ত্রণ কাঠামোর ধারণাটায়। নিয়ন্ত্রণ কাঠামোই তাই যা কম্পিউটারকে কম্পিউটার করেছে, প্রাককম্পিউটার সমস্ত হিশেবযন্ত্র থেকে আলাদা করে এনেছে। এর ভিতরেই ভরা থাকে সচেতন মানুষের চিন্তা, মানুষের তাই যন্ত্রের ইতিহাসে প্রথম। সচেতন মানুষের সারোগেট, প্রতিনিধি। যার অন্য নাম প্রোগ্রাম। আজকের আলোচনায় 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 কাঁচা ডেটাই থাকত মেশিনে। 'CC' থেকে ইন্সট্রাকশন ডেটাও থাকতে শুরু করল। তিন নম্বর দিনে কম্পিউটারের প্রাগ-ইতিহাসটা মনে করুন। ডিজিজড এঞ্জেল, ক্লিষ্ট দেবদূত, আডা লাভলেস, তার সেই স্বপ্নগুলো, যাকে আমরা এই শতাব্দীর সবচেয়ে উত্তেজক সায়েন্স ফিকশন বলে ডেকেছিলাম। মনে করুন কার্ডগুলোর কথা। কার্ডরা 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 পাঁচ রকম উপাদানের কথা মনে করুন। ইনপুট উপাদান, আউটপুট উপাদান, তথ্য প্রসেসিং-এর উপাদান, তথ্য হোল্ডিং-এর 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 প্রকৃতির উপর অন্য প্রাণীদের থেকে আলাদা করে দিল মানুষের ইতিহাস। কাঁচা প্রকৃতিকে যন্ত্র বানাতে শিখল মানুষ। ঠিক তাই করল পাঁচ নম্বর উপাদানটাও। 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, আডার ছিল। সকলেই কবি নয়, কেউ কেউ কবি। 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 লিখুন।

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

প্রথম লাইনটা ধ্রুবপদ, ব্যাশপুজোর মন্তর। দ্বিতীয় লাইনটাই কাজ, '/etc' ডিরেক্টরির ভিতর থেকে '`trash`' নামে ফাইল কপি করে আনা হোম ডিরেক্টরিতে। আর তৃতীয় লাইন ব্যাশবাণী, করেছি গো, করে ফেলেছি। স্ক্রিপ্টটাকে চালনীয় বানান, '`chmod +x trashcopy`'। এবার চালান, '`./trashcopy`'। কী পেলেন?

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

প্রথম লাইনটা এরর মেসেজ। কপি করা যায়নি, ওই নামের ফাইলই নেই ওই ডিরেক্টরিতে। বুদ্ধির পরাকাষ্ঠাটা পরের লাইনে। কান এঁটো করা হাসি সহ, '`Done`'। হায় গাধা, তুই তো কাজটা পারিসই নি, তাও '`Done`'। এবার ভাবা যাক, আসলে গাধাটা কে? আমরা তো ব্যাশকে না-গাধা হতে শেখাইনি। আডার বাক্যটা মনে করুন, এমন সব কাজই করতে পারবে মেশিন, যা কী করে করতে হয় তাকে আমরা বলে দিতে পারব। আমরা শেখাইনি, কী করে ডিসিশন নেবে ব্যাশ, কোথায় শুরু করবে, কোথায় থামবে। শেখাইনি কী করে ভুল সংশোধন করতে হয়। শেখাইনি, ফাইলটা আছে না নেই সেটা যাচাই করতে। এগুলো মেশিনের না, আমাদের কাজ। ক্যালকুলেটর বারবার টেপার পরেও যখন সাড়া পাইনা, ক্যালকুলেটরের বাইরে, এমনকি বাড়ির বাইরে যাই, ব্যাটারি কিনে আনি, লাগাই, আবার চলে মেশিন। সব নিয়ন্ত্রণই আমাদের। অথচ ক্রমে কতদূর নিয়ন্ত্রণ করা যাচ্ছে দেখুন, পরশুদিন তথাগত ফোন করে জানাল ও একটা নতুন ইউপিএস কিনেছে, যেটা সরাসরি তার ব্যাটারির বিদ্যুতের পরিমাণ কারনেলকে জানিয়ে দিতে পারে। গ্নু-লিনাক্স কারনেল অপশনে দেখবেন এইসব দেওয়া থাকে। ইউপিএস কখন চালু কখন বন্ধ হবে সেটাও এখন কারনেল দেখবে। এইসব মহার্ঘ ব্যাপার ছেড়ে দিন, আমাদের ফাইল কপি করার সমস্যাটা ভাবুন, সেই নিয়ন্ত্রণগুলো, আমাদের প্রয়োজনের দূরত্ব অব্দি, ব্যাশকে দিয়ে করানোই যায়। 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 আমরা নিজেরাই দেখি, বিচার করি। কপি কেন হলনা? দেখি তো ডিরেক্টরিটা, ফাইলটাই আদৌ আছে কিনা। এখন থেকে সেটা ব্যাশ করবে। প্রোগ্রামটার কাঠামোটা, ফ্লো-চার্টটা 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 সমস্ত টেকনিকাল খুঁটিনাটি বুঝতে আপনাকে আরো দুতিনটে প্যারা ওয়েট করতে হবে। স্ক্রিপ্টটা বানান 'trashcopy1' নামে একটা ফাইলে। তারপর তাকে 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
```

কাঠামোটা খেয়াল করুন। লুপটার গঠনটা হল 'if … then … else … fi'। 'if' দিয়ে শুরু, 'fi' দিয়ে শেষ। মধ্যে মধ্যে '#' চিহ্নটা দিয়ে যথারীতি মানবপাঠ্য মানববোধ্য মন্তব্য আনা হয়েছে, লাইনগুলো অব্যাশনীয়। মন্তব্যগুলো পড়ে দেখুন, সহজেই বুঝতে পারবেন, কী বলা এবং করা হয়েছে। শুধু দু-নম্বর লাইনটা বুঝতে 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'। আসলে এগুলো তো কাজের স্ক্রিপ্ট না, জাস্ট আপনাদের স্ক্রিপ্ট-পড়ার অভ্যেশ আনার জন্যে বানানো। আমার সব স্ক্রিপ্টই থাকে হোমের '~/bin' ডিরেক্টরিতে। তাই '.sh' না-থাকলেও বোঝা না-বোঝাটা 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'
```

আমাদের স্ক্রিপ্টও কথা বলতে শুরু করেছে, মহারাজ, আজ মনে হচ্ছে আমাদের বয়স হয়েছে। স্ক্রিপ্টটার শরীরে ইনডেন্টগুলো দেখুন, এক একটা কাজের ব্লককে ইনডেন্ট করে ডানে সরিয়ে এক এক সঙ্গে আনা হয়েছে, এটাও স্ক্রিপ্ট পড়ে বোঝার স্বার্থে। একটা ব্লক 'then' অংশটার, আর একটা ব্লক 'else' অংশটার। এবার ওই হিব্রু ভাষায় লেখা দু-নম্বর লাইনটা একটু মাপা যাক। এই লাইনে 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'। 'ফাইল' মানে যে কোনো ফাইল। কোনো 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। 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 বলেছিলাম, ওই ডিরেক্টরিতে ওই নামের কোনো ফাইল আছে কিনা সেটা যাচাই করতে। যদি না-থাকে সেটা আমাদের জানাতে বলাটা ছিল তার পরে। যাচাই করার কমান্ডটা ছিল, '`test -f /etc/trash`'। এই আদেশটা এর থেকে একটু অন্যরকম আকারেও দেওয়া যেত, আরো ছোট আকারে, '`[ -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
```

এটাও একই ভাবে চলবে। 'test' আদেশের এই আকারটা অনেক বহুলব্যবহৃত। 'test' প্রোগ্রামটা ব্যবহার হয় দুটো ভ্যারিয়েবল্, বা একটা ভ্যারিয়েবল এবং একটা কন্সটান্ট রাশির ভিতর তুলনায়। `[ "$number -eq 5 ]`' কমান্ড দিলে, 'test' যাচাই করবে 'number' ভ্যারিয়েবলের মান '5'-এর সমান কিনা। 'test' কমান্ডটা ব্যবহার করে এমন কেজো একটা স্ক্রিপ্ট ধরা যাক। এখানে যেটা দিচ্ছি সেটা আদত স্ক্রিপ্টটা নয়। তার সঙ্গে কিছু মন্তব্য এবং আরো কিছু যোগ করেছি, যাতে আপনার বুঝতে সুবিধে হয়। আমার মেশিনের তিনটে গ্নু-লিনাক্স ওএস-এর জন্যে স্ক্রিপ্টটা লেখা, সুজে, স্ল্যাকওয়ার, ম্যানড্রেক। কী নাম তাদের বা ঠিক তিনটেই কিনা তাতে কিছু এসে যায়না, দেখতেই পাবেন। প্রতিটা ওএস-এই একজন করে ইউজার 'dd' আছে। নানা সময়ে নানা কিছু করে দেখার জন্যে ব্যাশের লোকাল কনফিগারেশন ফাইলগুলোয় নানা বদল করি, সেই বদলগুলো মাঝে মাঝেই চূড়ান্ত ফ্লপ করে, তখন সামলানোর জন্যে 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 লাইন বাই লাইন বোঝা যাক। স্ক্রিপ্টে যে লাইনে মন্তব্য আছে তাদের বাদ দিয়েছি। ওই মন্তব্য পড়ে বুঝে গেলে এই লাইন ধরে ধরে আলোচনাটা পড়ার দরকারই নেই।

০১। ধ্রুবপদ, ব্যাশস্ক্রিপ্টের বাধ্যতা।

০২। '/mnt/arkive/bkp/bash' ডিরেক্টরি থেকে কনফিগারেশন ফাইলের পুরোনো ব্যাকআপ ফাইলগুলো উড়িয়ে দিতে বলা হচ্ছে ব্যাশকে।

০৪। স্ক্রিপ্ট জুড়ে যে কাউন্টারটা ব্যবহার করব, তাকে ০ করে নিচ্ছি, পুরো প্রক্রিয়ার ধাপগুলো যাতে গুনতে গুনতে যাওয়া যায়, যা থেকে তৈরি সূচক পরে ব্যাকআপ করা ফাইলগুলোর নামে ব্যবহার করব। সিস্টেম থেকে সিস্টেমে ফাইলগুলোকে আলাদা করার জন্যে। এই মুহূর্তে এই 'counter' ভ্যারিয়েবলের মান মানে '$counter' হল '1'। লুপের ধাপে ধাপে বাড়বে, স্ক্রিপ্টের ৩৯ নম্বর লাইনে গিয়ে। প্রতিবার বাড়বে এক করে।

০৬। শুরু হচ্ছে প্রথম 'for' লুপের নিয়ন্ত্রণ কাঠামো। তিনটে ডিরেক্টরির নাম দেওয়া আছে, '/', '/mnt/slackware', এবং '/mnt/mandrake'। মানে যেখানে যেখানে আমার মেশিনের তিনটে গ্নু-লিনাক্স ওএস-এর তিনটে রুট পার্টিশন মাউন্ট হয়। এই ব্যাশস্ক্রিপ্টটা বানানো সুজে সিস্টেমের ভিতরে দাঁড়িয়ে। সুজেতে '/mnt' ডিরেক্টরিতে দুটো সাবডিরেক্টরি '/mnt/slackware' আর '/mnt/mandrake' জুড়ে স্ল্যাকওয়ার আর ম্যানড্রেকের রুট পার্টিশন দুটো মাউন্ট হয়। এখানে দেওয়া ছকটা পাঠমালা জুড়ে ব্যবহৃত ছক থেকে একটু আলাদা, সেখানে কোনো ম্যানড্রেক ছিলনা। আগে তিনটে সিস্টেমের কথা আনতে চাইনি, জটিলতাটা বড্ড বেশি বেড়ে যাচ্ছিল বলে। এখন অনেকটা অভ্যস্ত হয়ে গেছি। এবং স্ক্রিপ্টটাকে আমার মেশিনের ব্যবস্থা থেকে একদম আলাদা ভাবেই পড়ুন। এমনকি সেই মেশিনে তিনটে ব্যবস্থা যদি নাও থাকে তাহলেও কিছু এসে যাবেনা। পরেরগুলো ব্যাশ আর পাবেনা, কিন্তু গ্নু-লিনাক্স ওএস থাকলে '/' থাকবেই, তার মানে অন্তত একবার স্ক্রিপ্টটা তার লুপে ঘুরবেই। এবার 'for' লুপটার কাঠামো খেয়াল করুন। আমরা 'directory' বলে একটা ভ্যারিয়েবল তৈরি করছি, যার তিনটে

আলাদা আলাদা মান হতে পারে '/', '/mnt/slackware', এবং '/mnt/mandrake'। 'for' লুপটা ব্যাশকে বলছে পরপর একটা করে '$directory' মানে 'directory' ভ্যারিয়েবলের মান তুলতে এবং সেই মান অনুযায়ী স্ক্রিপ্টের পরের লাইনের আদেশগুলো চালাতে। যেই একটা মানের জন্যে স্ক্রিপ্টের সব আদেশ পালন করা শেষ হবে, তখন পরের মানটা নিতে এবং আবার গোড়া থেকে শুরু করতে। আডার কার্ডের সঙ্গে মিল পাচ্ছেন?

১০। কাজ শুরু হচ্ছে। 'cd' করে সেই ডিরেক্টরিতে যাও যার মান তোমার কাছে এই মুহূর্তে ভরা আছে 'directory' ভ্যারিয়েবলের জমিতে। লুপের প্রথম ঘোরায় এই লাইনে এই মুহূর্তে '$directory' হল '/', তার মানে, '`cd /`' আদেশটা পালন করো। অর্থাৎ, এখন থেকে ব্যাশের ওয়ার্কিং ডিরেক্টরি হয়ে গেল '/'। লুপের দ্বিতীয় ঘোরায় এই লাইনে এসে সেটা হবে '/mnt/slackware', তৃতীয় ঘোরায় হবে '/mnt/mandrake'।

১২। এই লাইনটা মূল স্ক্রিপ্টে ছিলনা, আপনাদের জন্যে ঢোকানো, ছিপের ফাতনার মত, জলের তলার ঘটনার উপর নজরদারির একটা কৌশল। লুপের প্রতি ঘোরায় একবার করে ব্যাশ এই লাইনে আসবে এবং ফুটিয়ে তুলবে কোন ডিরেক্টরিতে আছে, তিনবার তিনটে আলাদা নাম আসবে। স্ক্রিপ্টটা যে ঠিক এগোচ্ছে সেটা বোঝা যাবে।

১৪। 'test' বিচার করে দেখছে, '$directory/home/dd' নামের কোনো ডিরেক্টরি আছে কিনা আদৌ। এই মুহূর্তে '$directory' হল '/', তার মানে ব্যাশ '/home/dd' নামের ডিরেক্টরি খুঁজছি। লুপের প্রতি ঘোরায় খোঁজাটা বদলাবে। দ্বিতীয় ঘোরায় খুঁজবে '/mnt/slackware/home/dd'। তৃতীয় ঘোরায়? লাইন ১৪-য় শুরু হচ্ছে আর একটা নিয়ন্ত্রণ কাঠামো, 'if'। এই স্ক্রিপ্টে 'if' কাঠামোটা তিনবার ব্যবহার হয়েছে, তার প্রথমবার এই শুরু হল।

১৬। লাইন ১৪-য় 'test' দিয়ে পরখ করে উদ্দীষ্ট ডিরেক্টরি পাওয়া গেলে সেটা জানানোর ব্যবস্থা। নির্বাক ব্যাশকে একটু কথা বলতে শেখানো। এই লাইনটাও মূল স্ক্রিপ্টে ছিলনা, আপনাদের জন্যে যোগ করা।

১৭। শুরু হলো দ্বিতীয় 'if'। প্রথম 'if' এখনো শেষ হয়নি, তার পেটের মধ্যে এই দ্বিতীয় 'if'। টেকনিকাল ভাষায় বলে নেস্টেড লুপ। লুপের বাচ্চা লুপ। অর্থাৎ, প্রথম 'if' যদি মেলে শুধু তবেই এই দ্বিতীয় 'if' কাঠামোয় ঢুকছি। প্রথম 'if'-এর শর্তটা না-মিললে আর দ্বিতীয় 'if'-এ যেতে হচ্ছেনা। '/home/dd' ডিরেক্টরি না-থাকলে দ্বিতীয় 'if' আর বিচার করার দরকারই হচ্ছেনা। দ্বিতীয় 'if' আবার একটা 'test', পরখ করছে '/home/dd' ডিরেক্টরিতে '.bashrc' ফাইল আছে কিনা। ভাবুন, ডিরেক্টরিটাই না-থাকলে এটা পরখ করার দরকারই পড়ত না, সবাই তো চেশায়ার বিড়াল নয় যে বিড়াল না-রইলেও বিড়ালের হাসিটা রয়েছে।

১৯। লাইন ১৭-য় 'test' করে ফাইলটা পাওয়া গেলে সেটা জানিয়ে দেওয়া।

২০। 'test' যদি না পায়, সেটাও বলে দিতে হয়, ব্যাশ, ছি, অত চুপ করে থাকা ভালো না।

২১। 'if'-এর শীর্ষাসন, মানে 'if' শেষ হল। কিন্তু ইফবহুল এই স্ক্রিপ্টের কোন 'if' ইনি? ঠিক এর আগেই যে 'if' শুরু হল। মানে দ্বিতীয় 'if'। 'fi' আমাদের জানাল, লাইন ১৭-য় বিসমিল্লা হওয়া দ্বিতীয় 'if' কাঠামোর ইন্তেকাল হল লাইন ২১-এ এসে।

২৩। শুরু হল লাইন ২৭ অব্দি স্ক্রিপ্টের তৃতীয় 'if' নিয়ন্ত্রণ। আর সব হুবহু এক দ্বিতীয় 'if'-এর সঙ্গে, শুধু এবার 'test' দিয়ে ব্যাশ খুঁজছে '.profile', এবং তার সংবাদ জানিয়ে দিচ্ছে লাইন ২৫ এবং ২৬-এ। লাইন ২৭-এ শেষ হচ্ছে এই তৃতীয় 'if'। প্রথম 'if' কিন্তু এখনো চলছে। তৃতীয় 'if'-ও নেস্টেড ইফ। প্রথম 'if' আবার চলছে প্রথম 'for' লুপের ভিতর। প্রথম 'for' থেকে এখনো বেরোইনি, ঢুকেছিলাম লাইন ০৬-এ।

২৯। ব্যাশ আবার ওয়ার্কিং ডিরেক্টরি বদলাচ্ছে। লাইন ১০-এ ব্যাশ ঢুকেছিল '/' ডিরেক্টরিতে। তারপর '/home/dd' ডিরেক্টরির ফাইলের খবর নিল সেখানে বসেই। এবার ঢুকছে '/home/dd'-তে, সেটাই এখন ওয়ার্কিং ডিরেক্টরি। এটা প্রথম 'for' লুপের প্রথম ঘোরা।দ্বিতীয় ঘোরায় লাইন ২৯-এ ব্যাশ ঢুকবে '/mnt/slackware/home/dd' ডিরেক্টরিতে। তৃতীয় ঘোরায় এই লাইন ২৯-এ এসে ব্যাশ কোন ডিরেক্টরিতে ঢুকবে?

৩১। এটা আমাদের চেনা কমান্ড সাবস্টিটিউশন। '/home/dd' ডিরেক্টরিতে '.bashrc' বা '.profile' ফাইল যা 'ls' করে পাওয়া যাবে, তাদের নিয়ে শুরু হচ্ছে এই দ্বিতীয় 'for' লুপ। এটাও নেস্টেড লুপ, প্রথম 'for' এখনো চলছে, চলছে প্রথম 'if' নিয়ন্ত্রণও। প্রথম 'for' লুপের ঘোরার সম্ভাব্য সর্বোচ্চ সংখ্যা তিন। এই দ্বিতীয় 'for'

লুপের ঘোরার সম্ভাব্য সর্বোচ্চ সংখ্যা দুই, কারণ ফাইল থাকতে পারে দুটো। যদি একটা ফাইলও না-থাকে, তাহলে তো ল্যাটা চুকেই গেল, লুপ পয়দা হয়েই মরে গেল, ব্যাশ এগিয়ে যাবে স্ক্রিপ্টের পরবর্তী লাইনে। দেখুন, এই দ্বিতীয় 'for' লুপে ভ্যারিয়েবলের নাম হল 'file'। '$file' বা 'file' ভ্যারিয়েবলের সম্ভাব্য মান মাত্র দুটো, হয় '.bashrc' নয় '.profile'। ধরুন, দুটো ফাইলই পেয়ে গেল ব্যাশ। সেই অবস্থায়, এই দ্বিতীয় 'for' লুপের প্রথমবার ঘোরার সময় '$file' বা 'file' ভ্যারিয়েবলের মান অবশ্যই '.bashrc'। দ্বিতীয়বার ঘোরার সময় '$file' দাঁড়াবে '.profile', যদি দুটো ফাইলই পাওয়া যায়।

৩৩। মজা দেখুন, গোটা স্ক্রিপ্ট জুড়ে ব্যাশ যে কাজটা করবে সেটা আসলে এই লাইন ৩৩-ই। যদি '.bashrc' আর '.profile' ফাইল দুটো পায়, বা তাদের একটাও পায়, সেই দুটো বা একটা ফাইল ব্যাশ এবার কপি করে দেবে '/mnt/arkive/bkp/bash' ডিরেক্টরিতে, যা অবশ্যই আগে থেকে বানানো আছে সিস্টেমে। ফাইল যদি ব্যাশ না-ই পায় আদৌ, সেটা তো আগেই জানিয়ে দিয়েছে। এবং দেখুন, কপি করার সময় ফাইলটার নাম বদলে দেবে। বদলের ছকটা হল 'bkp$file.$counter'। মানে, নামটার আগে একটা 'bkp', পরে '.$counter'। খেয়াল করুন, এখন '$counter' 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, তার সিগনাল, 'done'। প্রথম 'for' লুপ কিন্তু এখনো চলছে। এবং প্রথম 'if' নিয়ন্ত্রণও।

৩৬। এটা প্রথম 'if' নিয়ন্ত্রণের শেষ অংশ। লাইন ১৪-য় শুরু হওয়া 'if' নিয়ন্ত্রণের ভিতর আমরা এতটা সময় ধরে এগিয়েছি এই সম্ভাবনাটা ধরে যে, '/home/dd' ডিরেক্টরিটা পাওয়া গেছে। কিন্তু যদি ডিরেক্টরিটা না পাওয়া যায় আদৌ, তাহলে কী হবে? এবার সেই অন্য সম্ভাবনাটা। ব্যাশ আমাদের জানিয়ে দিচ্ছে, ডিরেক্টরিই নেই, ফাইল কপি করব কী করে?

৩৭। মানে প্রথম 'if' নিয়ন্ত্রণ শেষ হল এই লাইনে এসে, তারই সিগনাল 'fi'।

৩৯। এই কলকব্জাটা নিয়ে বিশদ আলোচনা আছে আগেই, এক করে বাড়িয়ে দিচ্ছি 'counter' ভ্যারিয়েবলটাকে। তার মানে, এবার '$counter' হয়ে গেল '2'। প্রথম 'for' লুপ দ্বিতীয় ঘোরায় তাই ব্যাক-আপ ফাইলে যোগ হবে '2'। যেখানে আগেরবার হল '1'। লাইন ৩৩-এর সঙ্গে মেলান। দেখুন, সুজের বেলায় হবে '1', স্ল্যাকের বেলায় '2', এবং ম্যানড্রেকের বেলায় '3'। যদি না পাওয়া তাহলে ব্যাক-আপ করতেই হবেনা।

৪১। এটাও বোঝার সুবিধের জন্যে লাগানো।

৪৩। এতটা পথ পেরোলে লুপ খতম বলা যায়? প্রশ্নটা খুব সহজ এবং উত্তর-ও তো জানা। কিন্তু, তরঙ্গ মিলায়ে যায়, তরঙ্গ উঠে, 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 এই তীব্র পিচদ্রব নিদাঘের পথে সিভিল ইঞ্জিনিয়ার আর আওয়ারা কুত্তাদের সঙ্গে একটা মোলায়েম 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 স্ক্রিপ্ট পড়ে অনেকটাই অভ্যস্ত হয়ে এসেছি। এই স্ক্রিপ্টটা নাম দিয়ে লিখে চালনীয় বানিয়ে ফেলুন। আমি নাম দিয়েছিলাম 'whileshow'। যে নামই দিন, সেটাকে লিখুন এবং চালান। আপনাদের একটু ঘাবড়ানো দরকার।

```
#!/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 নিরবচ্ছিন্ন ধারাপাত থেকে মুক্তি পাওয়ার একমাত্র উপায় '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 যে প্রসেসকে দিয়ে 'true' প্রোগ্রামটা ডেকে আনা হয়েছে সেই প্রসেসটাই মরে যায় বা মেরে ফেলা হয়। এখানে আমরা প্রসেসটাকে মেরে ফেলছি 'Ctrl-C' টাইপ করে। 'true'-এর ম্যানুয়াল পাবেন ম্যান পেজের সেকশন ১-এ। 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 হয়ে গেল, এবার নিজেই বোঝার চেষ্টা করুন, কী ঘটছে এখানে। ধাপে ধাপে আমরা বাড়াচ্ছি কাউন্টার মানে 'c' ভ্যারিয়েবলটাকে। আর সেটাকে ফুটিয়ে তুলছি স্ক্রিনে। প্রতিবার ফুটিয়ে তোলার পর 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' অর্থাৎ, '10'-এর ছোট বা সমান কিনা, লেস বা ইকোয়াল কিনা, সেটা পরখ করে নিচ্ছে ব্যাশ। 'test' কমান্ডের ম্যানুয়াল পড়ুন, সবগুলো খুঁটিনাটিই পেয়ে যাবেন। কাউন্টার ভ্যারিয়েবলটাকে আমরা জোড়াকোটের ভিতরে দিয়েছি। এটা বাধ্যতামূলক নয়, কিন্তু সুবিধাজনক, অনেক গোলযোগ কম হয়। এবার স্ক্রিপ্টটাকে চালান। পাবেন নিচের এই লাইনগুলো। কিন্তু প্রতিটা লাইন আসবে পরের লাইনের এক সেকেন্ড পরে। বেশ মজা লাগে চালিয়ে।

```
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। রেগুলার ইন্টারভালে ঘুম পাড়াচ্ছি আর জাগাচ্ছি। এক সেকেন্ড ঘুমোতে দেওয়া মানেই তো এক সেকেন্ড পরে জাগানো। এর 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? লুপটাকে খেয়াল করুন, লুপের শব্দগুলোকে, হোয়াইল (while) মানে 'যতক্ষণ অব্দি', আর আনটিল (until) মানে 'যতক্ষণ-না'। যতক্ষণ অব্দি দশের ছোট বা সমান, তার মানেই, যতক্ষণ-না দশের বড় বা সমান। এখন বাকি রয়েছে 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
```

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 দুহাজার তিন, থেকে ধরলে, আজ তেইশে জুন, দুহাজার চার, তেইশবাইশে মার্চ, মানে দুশো পঁয়ত্রিশ দিনের এই ম্যারাথন খতম। যদিও, যেমন বলেছিলাম, আপনাকে একটু গালাগাল দেওয়ার ব্যাপার আছে, সেটা আসছে ছোট হরফে, প্রতি দিনের শেষেই যেমন এসেছে। এতটা সময় লাগত না, দুটো খসড়া, তাদের সংশোধন ইত্যাদি মিলিয়ে অন্তত আরো পঁয়তাল্লিশ দিন কম লাগত। এর মধ্যে বেশ কয়েকবার আমার হার্ডডিস্ক গেছে, মাদারবোর্ড গন্ডগোল

করেছে, এসএমপিএস গন্ডগোল করেছে, ডেটা কেবল বদলাতে হয়েছে। এখন তো কাজ চলছে অন্যের এসএমপিএস এনে। আর হার্ডওয়ারের লোক ডাকার খরচ প্রায়ই পোষানো যায়না, এগুলো নিজেকেই করে নিতে হয়েছে, তাতে সময়ও বেশি নষ্ট হয়েছে। একবার হার্ডডিস্ক যাওয়ার উল্লেখ পাঠমালার মধ্যেও আছে মনে হচ্ছে।

প্রথমে বলি একটা ভালো কথা। শুধু পাঠমালার না, আমার জন্যেও ভালো। আজকাল ভালো কিছু এত কম দেখি, একটু কিছু দেখলেই মনটা ভারি ভরে যায়। এটা লিখে চলা আমায় একটা অন্য অনুভব দিয়েছে। আত্মীয়তার অন্য একটা মাত্রা। পাঠমালাটার জন্যে কী পরিমাণ খাটনি গেছে সেটা আন্দাজ করতে পারছেন। খাটনিটা বোধহয় দিতেই পারতাম না এই অভিজ্ঞতাটাকে বাদ দিয়ে। লেখা এবং পড়া — গত বেশ কিছু বছর মোটামুটি ভাবে এটাই আমার কাজ। লেখার প্রক্রিয়াটা সেখানে খুবই রোজকার একটা অভিজ্ঞতা, নিজে না হোক অন্যে, আমার চারপাশে নিয়তই লেখা হচ্ছে, এই লেখা বা ওই লেখা, এ লিখছে বা ও লিখছে। কিন্তু এতগুলো বছরের সেই অভিজ্ঞতাতে এরকম কিছু আমি কখনো দেখিনি। না আমার বেলায়, না অন্য-কারুর। অনুভবটা একটা সামগ্রিকতার। পাঠমালাটা লিখেছি শেষ অব্দি আমি, আমার মেশিনে, আমার কিবোর্ডে। আমি যদি এর 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 পাঠমালাটা বিয়োনোর কাজে পার্ট নিয়েছে, সেটা আমি কোনোদিন কারোর লেখার বেলাতেই করিনি, করার কথা ভাবিওনা, আসলে আমি ওদের চেয়ে অনেকটা বুড়ো হয়ে গেছি, তাই হয়তো মনও ছোট হয়ে গেছে আমার। এবার, যে গালাগালটা আপনাদের দেওয়ার, তার সঙ্গে এর একটা খুব নিকট সম্পর্ক। এই যত্নের প্রক্রিয়ার মধ্যেই ওরা সেই সবই করেছে পাঠমালাটার 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 এখানেই। আজ যদি এর অন্য ভাষায় অনুবাদ হয়, ইংরিজি হলে তো বটেই, মলয়ালম হিন্দি তামিল ইত্যাদির বেলাতেও, একটা অপমান তাদের মেনে নিতে হবেনা, পাঠমালাটা লেখার একদম গোড়া আমাকে যা মেনে নিতে হয়েছিল। যদি গোড়া থেকেই ইংরিজিতে লিখতাম তাহলেও সেটা হতনা। 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 সিরিয়াস লেখালেখির উপায় নেই। যা আছে তাতে এত বড় একটা লেখা শুধু অতুলনীয় রকমের কষ্টসাধ্য তাই নয়, বেশ কিছুটা বোকামিও। এত বেশি নিজেকে অপব্যয় করতে হবে সেইসব উদ্ভট প্রক্রিয়ায়। অথচ, ইংরিজি তো ছেড়েই দিন, তামিল মলয়ালম হিন্দিতে আছে। কেন? আপনার জন্যে। আপনিও যদি থাকতেন এই পাঠমালায়, 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।

glt-mad@ilug-cal.org