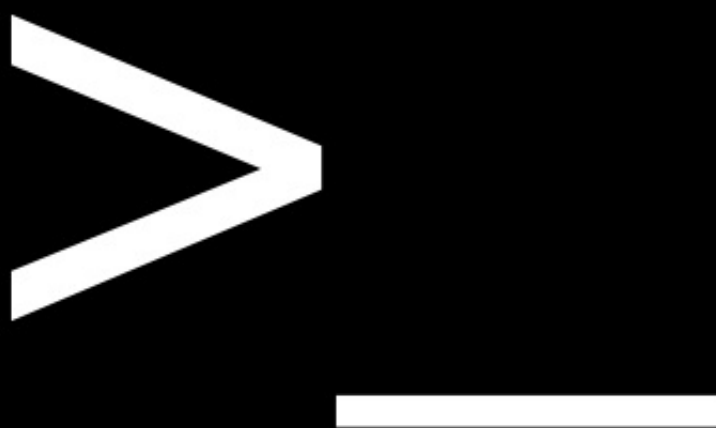


# শেল ও শেলস্ক্রিপ্টিং

[sh.howtocode.com.bd](http://sh.howtocode.com.bd)



[howtocode.com.bd](http://howtocode.com.bd)

# সূচিপত্র



শুরুর কথা	0
প্রথম খন্ড - শেল বেসিক	1
প্রথম অধ্যায় - শেল বেসিক	1.1
শেল ও প্রম্পট	1.1.1
কমান্ড	1.1.2
নেভিগেশন	1.1.3
আরো একটু ls	1.1.4
ফাইল	1.1.5
লিঙ্ক	1.1.6
লিনাক্স ফাইলসিস্টেম	1.1.7
দ্বিতীয় অধ্যায় - ম্যানিপুলেশন	1.2
ওয়াইল্ডকার্ড	1.2.1
ফাইল ও ডিরেক্টরি তৈরি করা	1.2.2
ফাইল ও ডিরেক্টরি কপি করা	1.2.3
ফাইল ও ডিরেক্টরি মুভ করা	1.2.4
ফাইল ও ডিরেক্টরি রিমুভ করা:	1.2.5
হার্ডলিঙ্ক ও সফ্টলিঙ্ক তৈরি করা	1.2.6
অনুশীলন	1.2.7
তৃতীয় অধ্যায় - রিডিরেকশন	1.3
স্ট্যান্ডার্ড ইনপুট, আউটপুট এবং এরর	1.3.1
স্ট্যান্ডার্ড আউটপুট রিডিরেকশন	1.3.2
স্ট্যান্ডার্ড এরর রিডিরেকশন	1.3.3
স্ট্যান্ডার্ড আউটপুট ও এরর একত্রে রিডিরেকশন	1.3.4
ফাইল সংযুক্তিকরণ	1.3.5
পাইপলাইন	1.3.6
চতুর্থ অধ্যায় - শেলের চোখে দেখা	1.4
এক্সপ্যানসন	1.4.1
পাথনেম এক্সপ্যানসন	1.4.2
গাণিতিক এক্সপ্যানসন	1.4.3
ব্রেস এক্সপ্যানসন	1.4.4

প্যারামিটার এক্সপ্যানসন	1.4.5
কমান্ড সাবস্টিটিউশন	1.4.6
ক্যাটিং	1.4.7
স্কেইপিং ক্যারেটার	1.4.8
পঞ্চম অধ্যায় - কীবোর্ড ট্রিক্স	1.5
কমান্ডলাইন এডিটিং	1.5.1
কমপ্লিশন	1.5.2
কমান্ড হিস্ট্রি	1.5.3
ষষ্ঠ অধ্যায় - পারমিশন	1.6
ওনার, গ্রুপ এবং অন্যান্য	1.6.1
এক্সেস রাইট	1.6.2
ফাইল পারমিশন পরিবর্তন	1.6.3
ফাইল পারমিশন মাস্কিং	1.6.4
বিশেষ পারমিশন	1.6.5
ওনার ইউজার ও গ্রুপ পরিবর্তন	1.6.6
পরিচয় পরিবর্তন	1.6.7
পাসওয়ার্ড পরিবর্তন	1.6.8
সপ্তম অধ্যায় - প্রসেস	1.7
প্রসেস এর প্রাথমিক ধারণা	1.7.1
প্রসেস দেখা	1.7.2
প্রসেস নিয়ন্ত্রণ	1.7.3
সিগন্যাল	1.7.4
দ্বিতীয় খন্ড - কনফিগারেশন ও এনভায়রনমেন্ট	2
প্রথম অধ্যায় - এনভায়রনমেন্ট	2.1
এনভায়রনমেন্টের ভিতরে দেখা	2.1.1
যেভাবে এনভায়রনমেন্ট তৈরী হয়	2.1.2
এনভায়রনমেন্ট পরিবর্তন	2.1.3
দ্বিতীয় অধ্যায় - প্রস্পট সম্পাদনা	2.2
প্রস্পট কাস্টমাইজেশন	2.2.1
প্রস্পট রঙ করা	2.2.2
কার্সরের অবস্থান পরিবর্তন	2.2.3
তৃতীয় খন্ড - আটপোরে কমান্ডলাইন	3
প্রথম অধ্যায় - প্যাকেজ ম্যানেজমেন্ট	3.1
দ্বিতীয় অধ্যায় - টেক্সট এডিটর	3.2

ন্যানো	3.2.1
ন্যানোর প্রাথমিক ব্যবহার	3.2.1.1
ন্যানো - এডিটিং এবং নেভিগেশন	3.2.1.2
ন্যানো কনফিগারেশন	3.2.1.3
ভিম	3.2.2
ভিম-এর এডিটিং মোড	3.2.2.1
ভিম-এর বেসিক এডিটিং	3.2.2.2
ভিম: সার্চ এ্যান্ড রিপ্লেস	3.2.2.3
ভিম: একাধিক ফাইল নিয়ে কাজ করা	3.2.2.4
ইম্যাকস্	3.2.3
ইম্যাকস্: প্রথম ধাপ	3.2.3.1
ইম্যাকস্: ক্যারেটার, কী এবং কমান্ড	3.2.3.2
ইম্যাকস্: বেসিক এডিটিং	3.2.3.3
ইম্যাকস্: সার্চ এ্যান্ড রিপ্লেস	3.2.3.4
ইম্যাকস্: একাধিক ফাইল এডিট করা	3.2.3.5
তৃতীয় অধ্যায় - স্টোরেজ মিডিয়া	3.3
লিনাক্সের চোখে স্টোরেজ ডিভাইস	3.3.1
মাউন্ট এবং আনমাউন্ট	3.3.2
পার্টিশন এবং ফরম্যাট করা	3.3.3
ফাইলসিস্টেম টেস্ট এবং রিপেয়ার করা	3.3.4
ডিভাইস ক্লোনিং	3.3.5
ইমেজ তৈরী	3.3.6
অপটিক্যাল মিডিয়ায় রাইট করা	3.3.7
চতুর্থ অধ্যায় - নেটওয়ার্কিং	3.4
নেটওয়ার্ক পরীক্ষণ এবং পর্যবেক্ষণ	3.4.1
ফাইল ট্রান্সফার	3.4.2
নিরাপদ যোগাযোগ	3.4.3
পঞ্চম অধ্যায় - ফাইল সার্চ	3.5
Locate: নাম দিয়ে ফাইল সার্চ	3.5.1
find: শক্তিশালী সার্চ	3.5.2
find: টেস্ট	3.5.2.1
find: অপারেটর	3.5.2.2
find: একশন	3.5.2.3

find: অপশন	3.5.2.4
অনুশীলন	3.5.2.5
ষষ্ঠ অধ্যায় - আর্কাইভ ও ব্যাকআপ	3.6
ডাটা কম্প্রেশন	3.6.1
ডাটা আর্কাইভিং	3.6.2
dtrx	3.6.3
সিনক্রোনাইজেশন	3.6.4
সপ্তম অধ্যায় - আটপোরে টুলস	3.7
রেঞ্জার(Ranger): ফাইল ম্যানেজার	3.7.1
মুট (Mutt): ইমেইল ক্লায়েন্ট	3.7.2
সিমিউজ(cmus): মিউজিক প্লেয়ার	3.7.3
ইলিন্কস (elinks): ওয়েব ব্রাউজার	3.7.4
উইচ্যাট (weechat) : আইআরসি ক্লায়েন্ট	3.7.5
ফিঞ্চ (finch) : চ্যাট ক্লায়েন্ট	3.7.6
অষ্টম অধ্যায় - প্রোগ্রাম কম্পাইলেশন	3.8
চতুর্থ খন্ড - টেক্সট ম্যানিপুলেশন	4
প্রথম অধ্যায় - রেগুলার এক্সপ্রেশন	4.1
গ্রেপ (grep)	4.1.1

# শেল ও শেলস্ক্রিপ্টিং

 Like  Share 9.6K people like this. [Sign Up](#) to see what your friends like.

স্বয়ংক্রিয় কন্ট্রিবিউটরের তালিকা  
(প্রথম ৫ জন)

- [\[248\] উৎসব রায়\(Utsob Roy\)](#)
- [\[007\] Md. Sabbir Alam](#)
- [\[006\] Nuhil Mehdy](#)
- [\[002\] Ashfaqur Rahman](#)

## সংক্ষেপ

লিনাক্স শেল এবং শেলস্ক্রিপ্টিং এর প্রাথমিক পাঠ এটি। একটি ওপেনসোর্স প্রোজেক্ট বলে আশা করা যাচ্ছে বাঙলায় লিনাক্স শেল এর সকল খুঁটিনাটি ক্রমে যুক্ত হবে।

## ওপেন সোর্স

এই বইটি মূলত স্বেচ্ছাশ্রমে লেখা এবং বইটি সম্পূর্ণ ওপেন সোর্স। এখানে তাই আপনিও অবদান রাখতে পারেন লেখক হিসেবে। আপনার কন্ট্রিবিউশান গৃহীত হলে অবদানকারীদের তালিকায় আপনার নাম যোগ করে দেওয়া হবে।

এটি মূলত একটি [গিটহাব রিপোজিটোরি](#) যেখানে এই বইয়ের আর্টিকেল গুলো মার্কডাউন ফরম্যাটে লেখা হচ্ছে। রিপোজিটোরি ফর্ক করে পুল রিকুয়েস্ট পাঠানোর মাধ্যমে আপনারাও অবদান রাখতে পারেন।

## শুরুর কথা

গ্রাফিক্যাল কম্পিউটিং এর দুনিয়ায় কমান্ডলাইন শিখতে যাওয়া প্রাথমিকভাবে হাস্যকর মনে হতে পারে। এটা সত্যি আপনি কমান্ডলাইনে ফেসবুক ব্যবহার করতে পারবেন না। ভিডিও দেখতে পারবেন না দেখার মতই এবং গ্রাফিক্স ডিজাইনও সম্ভব হবে না সহজে।

কিন্তু, লিনাক্স কমান্ডলাইন বা টার্মিনাল শেখাটা আপনার লিনাক্স ব্যবহারের অভিজ্ঞতার কিছু স্থায়ী পরিবর্তন আনবে। সম্ভবত প্রথমবার আপনি বুঝতে পারবেন পাওয়ার ইউজার হওয়ার আসল অর্থটি কি।

আসুন, আপনাকে এর ক্ষমতার একটা ছোট উদাহরণ দিই। একবার আমার ২শতাধিক ছোট ছোট লেখা মাইক্রোসফট ওয়ার্ড ডকুমেন্ট থেকে প্লেইন টেক্সটে কনভার্ট করার দরকার পড়েছিল। আমি হয়ত প্রত্যেকটা ফাইল খুলে সেখান থেকে টেক্সট ফরম্যাটে সেভ দিতে পারতাম। কিন্তু টার্মিনালে মাত্র একটি কমান্ডের মাধ্যমে আমি মাত্র কয়েকসেকেন্ডে

সবগুলো ফাইল কনভার্ট করেছি।

## আপনার যা যা প্রয়োজন হবে:

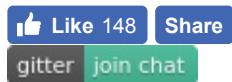
এই কোসটি কম্পিল্ট করতে আপনার একটি লিনাক্সভিত্তিক অপারেটিং সিস্টেম প্রয়োজন হবে। আমি এটা লেখা শুরুর সময়ে উবুন্টু জিনোম ১৪.০৪ ও পরবর্তীতে আর্চ লিনাক্স ব্যবহার করেছি।

\*\*কোনো কোনো ক্ষেত্রে আমি টার্মিনালেই সরাসরি বাঙলা ব্যবহার করেছি। এটা জরুরি কিছুই নয়। অনেক টার্মিনাল ইমুলেটরের বাঙলা ফন্ট রেন্ডারিং হতাশাজনক। তবে আপনি চাইলে **konsole** ও **konsole**নির্ভর ড্রপডাউন টার্মিনাল ইমুলেটর ব্যবহার করতে পারেন।

## কৃতজ্ঞতা

- উইলিয়াম শটস জুনিয়র ('দি লিনাক্স কমান্ডলাইন' বইটির লেখক।)
- রিচার্ড এম স্টলম্যান ('জিএনইউ ইম্যাকস ম্যানুয়াল' এর জন্য।)

তাহলে আসুন, শুরু করা যাক!



This work is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License](#).

## প্রথম খন্ড

### শেল বেসিক

এই অংশের লেখাগুলো একদম বেসিক শেল ব্যবহারের উপর লেখা হচ্ছে। আমরা জানতে চেষ্টা করবো টার্মিনালের বেসিক ব্যবহার, ফাইল ম্যানিপুলেশনের ম্যাজিক, রিডিরেকশনের সৌন্দর্য। দেখবো শেল যেভাবে দেখে, জানবো কিছু শর্টকাট যা টার্মিনাল ব্যবহারকে করবে স্মুথ। জানবো পারমিশন ও প্রসেস সম্পর্কে।

- অধ্যায় - এক: প্রাথমিক জানাশোনা
- অধ্যায় - দুই: ম্যানিপুলেশন
- অধ্যায় - তিন: রিডিরেকশন
- অধ্যায় - চার: শেলের চোখে দেখা
- অধ্যায় - পাঁচ: কীবোর্ড ট্রিক্স
- অধ্যায় - ছয়: পারমিশন
- অধ্যায় - সাত: প্রসেস



## অধ্যায় - এক

# প্রাথমিক জানাশোনা

- **শেল ও প্রম্পট:** শেল সম্পর্কিত ধারণা, টার্মিনাল পরিচিতি ও প্রম্পটের ধারণা।
- **কমান্ড:** কমান্ড এর ধারণা ও বিস্তারিত
- **নেভিগেশন:** ফাইলসিস্টেমে ঘোরাঘুরি, **pwd**, **ls** ও **cd** কমান্ডের ব্যবহার।
- **আরো একটু ls:** **ls** এর উচ্চতর ব্যবহার।
- **ফাইল:** ফাইল সংক্রান্ত ধারণা ও **file** এবং **less** কমান্ডের ব্যবহার।
- **লিঙ্ক:** লিঙ্ক এর ধারণা।
- **লিনাক্স ফাইলসিস্টেম:** ফাইলসিস্টেমের প্রাথমিক ধারণা।

# টার্মিনাল পরিচিতি

## শেল কি?

আমরা যখন কমান্ডলাইনের কথা বলি তখন আমরা আসলে শেলের কথা বলি। শেল হচ্ছে একটি ছোট প্রোগ্রাম যা কীবোর্ড থেকে কমান্ড অপারেটিং সিস্টেমকে পৌঁছে দেয়।

একসময় স্টিভ বর্ন নামের একজন প্রোগ্রামার sh নামে একটি শেল প্রোগ্রাম লেখেন ইউনিক্স অপারেটিং সিস্টেমের জন্য। পরে GNU প্রজেক্ট bash নামে তারই একটি উন্নত সংস্করণ তৈরী করে। যার পুরো নাম "Bourne Again Shell"। এখনকার প্রায় সব লিনাক্স অপারেটিং সিস্টেমে bash ডিফল্টভাবে দেয়া থাকে।

## টার্মিনাল ইমুলেটর

যখন আমরা গ্রাফিকাল ইউজার ইন্টারফেস ব্যবহার করি তখন শেলকে ব্যবহার করতে আমাদের আরেকটি প্রোগ্রাম প্রয়োজন পড়ে। টার্মিনাল ইমুলেটর বা টার্মিনাল। সব গ্রাফিকাল ইন্টারফেসওয়ালা লিনাক্স ডিস্ট্রিবিউশনে একাধিক টার্মিনাল ইমুলেটর(এখন থেকে আমরা টার্মিনাল ইমুলেটর না বলে সংক্ষেপে শুধু টার্মিনাল বলবো।) ইন্সটল করাই থাকে। যেমন KDE তে konsole বা GNOME এ gnome-terminal।

## প্রস্পট

আপনি যখন কোনো টার্মিনাল ওপেন করবেন এমনকিছু দেখবেন:

```
me@howtocode-pc:~$
```

দুর্য্যোগ লাগছে? ব্যাখ্যা করছি। আপনি হয়ত একইরকম দেখবেন না। ফরম্যাটটা একইরকম হবে অধিকাংশ লিনাক্স টার্মিনালে। ফরম্যাটটি হল `username@hostname:Working_directoryUserPrivilege`। আসুন, কাটাছেঁড়া করে দেখি।

- **username:** এটা হচ্ছে টার্মিনাল ব্যবহারকারীর ইউজারনেম। এখানে 'me'।
- **hostname:** হোস্টনেম হচ্ছে কম্পিউটারকে দেয়া একটি নাম। এই নামটি নেটওয়ার্কেও দেখায়। এখানে 'howtocode-pc'।
- **Working\_directory:** সহজ ভাষায় এটা হচ্ছে এখন এই শেলে আপনি কম্পিউটারের কোন ফোল্ডার বা ডিরেক্টরিতে আছেন। এখানে '~'। উল্লেখ্য, '~' এই চিহ্ন দিয়ে বর্তমান ইউজারের হোম ডিরেক্টরি বুঝায়।
- **UserPrivilege:** লিনাক্সে দুরকম ইউজার হয় সাধারণত। সাধারণ ব্যবহারকারী যার জন্য '\$' চিহ্ন এবং সর্বময় ক্ষমতার অধিকারী রুট ইউজার বা সুপারইউজার বা এডমিনিস্ট্রেটর যার জন্য '#' চিহ্ন।

তাহলে পুরো প্রস্পটটা কি বোঝাতে চাইছে? এটা বলতে চাচ্ছে, **me** নামের একজন সাধারণ ইউজার **howtocode-pc** নামের একটি কম্পিউটারে তার নিজের হোম ডিরেক্টরিতে কাজ করছে।

যখনি আপনি দেখবেন টার্মিনালে প্রম্পটের পর আপনার কার্সর আপনি বুঝবেন সে আপনার কমান্ড নিতে প্রস্তুত। টার্মিনাল যতক্ষণ কাজ করে নতুন কোনো প্রম্পট দেখায় না। কাজ শেষ করলে আবার প্রম্পটটি ফিরে আসে।

টার্মিনাল বন্ধ করতে লিখুন `exit`।

## টার্মিনালে কীবোর্ড ও মাউসের ব্যবহার

কী(key)	কমান্ড
Up arrow key	কমান্ড হিস্ট্রির পূর্ববর্তী কমান্ডটি প্রম্পটে নিয়ে আসে।
Down arrow key	কমান্ড হিস্ট্রির পরবর্তী কমান্ড প্রম্পটে নিয়ে আসে।
Left arrow key	প্রম্পটে কার্সর এক অক্ষর পিছনে আনে।
Right arrow key	প্রম্পটে কার্সর এক অক্ষর সামনে আনে।
Ctrl-c	টার্মিনালে চলতে থাকা কাজ সম্পূর্ণরূপে বন্ধ করে।
Ctrl-z	টার্মিনালে চলতে থাকা কাজ সাময়িকভাবে বন্ধকরে পরবর্তীতে চাইলে আবার চালু করা যাবে।
Ctrl-Shift-c	কপি।
Ctrl-Shift-v	পেস্ট।

## কমান্ড(Command)

কমান্ড শুনে হয়ত বুঝতেই পারছেন যে এর অর্থ আপনি শেলকে কিছু করতে নির্দেশ দেবেন। স্বাভাবিকভাবেই শেল আমাদের ভাষা বুঝবে না, তাকে বলতে হবে তার মত করে। আসুন কয়েকটা কমান্ড টাই করুন:

```
me@howtocode-pc:~$ date
Wed Aug 27 12:15:21 BDT 2014
```

```
me@howtocode-pc:~$ cal
August 2014
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

আমরা দুটো খুব সহজ কমান্ড দিয়েছি। আপনি 'date' লিখে এন্টার চাপুন। দেখবেন পরের লাইনে তারিখ ও সময় জানিয়ে দেবে। এবং তারপর আবার প্রম্পট ফেরত আসবে। আপনি তারপর 'cal' কমান্ডটি দিন, দেখবেন একটি ক্যালেন্ডার আসবে আজকের দিন হাইলাইট করে। একটা গুরুত্বপূর্ণ কথা বলে রাখি, আপনি আপ এ্যারো কী চেপে চেপে আগে দেয়া কমান্ডের হিস্ট্রি দেখতে পারবেন।

এইতো হল কমান্ড দেয়া শেখা। আসুন কমান্ডের কিছু নাড়িনক্ষত্রের খবর জেনে নিই।

### কমান্ডের রকমফের

আমরা যে কমান্ডগুলো ব্যবহার করি, সেগুলো আসলে কী? কমান্ডগুলোকে চারটি ভাগে ভাগ করা যায়:

- **Program:** কোনো প্রোগ্রাম(লিনাক্সে অধিকাংশ প্রোগ্রামই বেশকিছু কমান্ডলাইন ফাংশনালিটি দেয়।) আপনার কমান্ড হতে পারে। সেক্ষেত্রে তার নামটি কমান্ড হিসেবে ব্যবহৃত হয়। যেমন আপনি মুভি দেখার জন্য ডিএলসি প্লেয়ার ইন্সটল করলেন। টার্মিনালে vlc লিখে এন্টার চাপুন দেখবেন ডিএলসি ওপেন হবে। আমরা উপরে যে date ও cal কমান্ডদুটি ব্যবহার করেছি তাও একটি প্রোগ্রাম।
- **Shell Builtin:** শেল প্রোগ্রামগুলোর কিছু বিল্টইন কমান্ড থাকে। এগুলোর আলাদাভাবে অস্তিত্ব থাকে না। এগুলোও কমান্ড। যেমন: type।
- **Shell Function:** কোনো কাজের জন্য কিছু কমান্ডকে বিশেষভাবে ব্যবহার করা হয়। কিন্তু যদি এমন হয় যে বারবার ওই কমান্ডগুলো ব্যবহার করতে হচ্ছে তখন সেগুলোকে একসাথে একটি ফাংশনে রূপ দেয়া হয় কম পরিশ্রমে কাজ করার জন্য। ফাংশনটির নাম দিয়ে ফাংশনটিকে ব্যবহার করা যায় এবং এটিও কমান্ড হিসেবে বিবেচিত হবে। ফাংশন নিয়ে এই বইতে পরে বিস্তারিত বলা হবে।
- **alias:** এলিয়াসকে সহজভাষায় ডাকনাম বলা যায়। একটা কমান্ডের বিশেষ ব্যবহারিক রূপের ডাকনাম দেয়া যেতে পারে নিজের সুবিধার্থে। এই ডাকনামগুলোও কমান্ড। মনে করুন আপনি `ls -lAh` কমান্ডটি ব্যবহার

করেন। বারবার পুরোটুকু লেখা বিরক্তিকর। তাই `alias ll='ls -lAh'` কমান্ড দিয়ে আপনি `ll` নামে একটি এলিয়াস তৈরী করে নিতে পারেন কমান্ডটির। তাহলে আর বারবার পুরোটুকু না লিখে `ll` লিখলেও হবে। এই `ll` ও কমান্ড হিসেবে বিবেচিত হবে।

## কমান্ডের অংশসমূহ

এই কমান্ডটি দেখুন:

```
me@howtocode-pc:~$ cp -rv /var/cache/apt/archive/ ~/backup/
```

এটাকে কাঁটাছেঁড়া করা যাক:

- **cp:** এটি কমান্ডের নাম, যে নামে কমান্ডটিকে ডাকা হয়। এই কমান্ডটি ফাইল ও ফোল্ডার কপি করতে ব্যবহৃত হয়।
- **-rv:** '-' বা '--' এই চিহ্ন দিয়ে কমান্ডের অপশনগুলো একটিভেট করা হয়। এখানে আসলে দুটো অপশন, `-r` এবং `-v`। দুটোকে একসাথে `-rv` লেখা হয়েছে। কমান্ডগুলো স্বাভাবিকভাবে যে কাজ করে তারচেয়েও বেশি কিছু করতে বলার জন্যই অপশন দেয়া। এখান `-r` দিয়ে নির্দেশ করা হচ্ছে ওই ফোল্ডারের ভিতরের সবকিছুসহ(recursive) যেন কপি করা হয় এবং `-v` দিয়ে বলা হচ্ছে কমান্ড চলাকালীন সময় বিস্তারিত(verbose) কি হচ্ছে যেন জানানো হয়।
- **/var/cache/apt/archives/ ~/backup/:** এই অংশটা আর্গুমেন্ট। যখন কপি করতে বলছি, তখন নিশ্চয়ই বলা উচিত কি কপি করবে এবং কোথায় কপি করবে। এইখানে স্পেস দিয়ে আলাদা করা দুটি আর্গুমেন্ট আছে। প্রথমটি কি কপি করতে হবে ও দ্বিতীয়টি কোথায় কপি করতে হবে তা বলে দেয়।

তাহলে পুরো কমান্ডটিতে আমরা কি বলতে চাচ্ছি? আমরা বলতে চাচ্ছি যে 'root' ফোল্ডারের ভেতর(কোনো ফাইল বা ডিরেক্টরির পাথের শুরুতে '/' থাকলে সেটাকে রুট বোঝায়) যে 'var' ফোল্ডার, তারমধ্যে 'apt' এবং তারমধ্যে 'archives' নামের যে ফোল্ডারটি তাকে তার ভিতরে রাখা সবকিছু সহ(recursively) হোম ফোল্ডারের মধ্যে 'backup' ফোল্ডারে কপি করে রাখতে হবে। এবং এই কাজের বিস্তারিত জানাতে হবে।

## কমান্ড চেনা, কমান্ড জানা

আমরা জেনেছি ৪ রকম কমান্ডের কথা, আর জেনেছি সেই কমান্ড কত দুর্বোধ্যভাবে ব্যবহার করা যায়। একটা প্রশ্ন আসতে পারে, বিপুল পরিমাণ কমান্ডের মধ্যে যদি নতুন কোনো কমান্ড চোখে পড়ে, তো কিভাবে জানবো তার কথা? এবার দেখা যাক কমান্ড চেনা ও জানার উপায়।

### কমান্ড চেনা

#### type

**type** কমান্ডটির মাধ্যমে আপনি জানতে পারবেন কোনো কমান্ড উপরোক্ত চার ধরনের কোনটি তা জানা যাবে। এজন্য এর আর্গুমেন্ট হিসেবে ওই কমান্ডটি দিতে হবে। অর্থাৎ **cp** কমান্ডটি কোনধরনের তা জানতে আপনার লিখতে হবে **type cp**। কয়েকটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ type cp
cp is /bin/cp
me@howtocode-pc:~$ type type
type is a shell builtin
me@howtocode-pc:~$ type ll
ll is aliased to `ls -aIF`
```

এখানে **cp** এর জন্য তার পথ দেখাচ্ছে অর্থাৎ এটি এক্সিকিউটেবল প্রোগ্রাম। অন্যদিকে **type** শেল বিল্টইন এবং **ll** এলিয়াসড করা আছে **ls -aIF**। অর্থাৎ **ll ls -aIF** এর ডাকনাম।

## which

কোনো কমান্ড যদি এক্সিকিউটেবল প্রোগ্রাম হয় তবে তার লোকেশন **which** দিয়ে জানা যায়। যেমন:

```
me@howtocode-pc:~$ which ls
/bin/ls
```

অর্থাৎ **ls** প্রোগ্রামটি রুট ফোল্ডারের মধ্যে bin ফোল্ডারে আছে।

## কমান্ড জানা

### help

bash এর একটি ফিচার হল এতে বিল্টইন কমান্ডগুলোর বিস্তারিত জানা যায় **help** কমান্ড দিয়ে। যেমন:

```
me@howtocode-pc:~$ help cd
cd: cd [-L|[-P [-e]] [-@]] [dir]
    Change the shell working directory.

Change the current directory to DIR.  The default DIR is the value of the
HOME shell variable.

The variable CDPATH defines the search path for the directory containing
DIR.  Alternative directory names in CDPATH are separated by a colon (:).
A null directory name is the same as the current directory.  If DIR begins
with a slash (/), then CDPATH is not used.

If the directory is not found, and the shell option `cdable_vars' is set,
the word is assumed to be a variable name.  If that variable has a value,
its value is used for DIR.

Options:
  -L    force symbolic links to be followed: resolve symbolic links in
DIR after processing instances of `..'
  -P    use the physical directory structure without following symbolic
links: resolve symbolic links in DIR before processing instances
of `..'
  -e    if the -P option is supplied, and the current working directory
cannot be determined successfully, exit with a non-zero status
  -@    on systems that support it, present a file with extended attributes
        as a directory containing the file attributes

The default is to follow symbolic links, as if `-L' were specified.
`..' is processed by removing the immediately previous pathname component
back to a slash or the beginning of DIR.

Exit Status:
Returns 0 if the directory is changed, and if $PWD is set successfully when
-P is used; non-zero otherwise.
```

## --help

অনেক এক্সিকিউটেবল প্রোগ্রাম **--help** অপশনটি সাপোর্ট করে যা দিয়ে ওই কমান্ডটি ব্যবহারের নিয়মকানুন জানা যায়। যেমন:

```
me@howtocode-pc:~$ mkdir --help
Usage: mkdir [OPTION]... DIRECTORY...
Create the DIRECTORY(ies), if they do not already exist.

Mandatory arguments to long options are mandatory for short options too.
-m, --mode=MODE    set file mode (as in chmod), not a=rwx - umask
-p, --parents       no error if existing, make parent directories as needed
-v, --verbose       print a message for each created directory
-Z, --context=CTX   set the SELinux security context of each created
                    directory to CTX
--help             display this help and exit
--version          output version information and exit

Report mkdir bugs to bug-coreutils@gnu.org
GNU coreutils home page: <http://www.gnu.org/software/coreutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
For complete documentation, run: info coreutils 'mkdir invocation'
```

## man

**man** manual এর সংক্ষিপ্ত রূপ। অধিকাংশ কমান্ডের বিস্তারিত খুঁটিনাটি জানতে এই কমান্ডটি অতুলনীয়। এমনকি সিস্টেমের অনেক গুরুত্বপূর্ণ ব্যাপারও এখানে আলোচনা করা থাকে। কোনো কমান্ডের ম্যানপেজ দেখতে হলে কমান্ডের নামটি এই কমান্ডের আর্গুমেন্ট হিসেবে দিতে হবে। যেমন **cp** এর ম্যানপেজের জন্য **man cp**। এই কমান্ড দেয়ার পর ম্যানপেজটি **less** নামের একটি কমান্ডলাইন টেক্সটভিউয়ারে খুলবে, আপনি এ্যারো কী দিয়ে স্ক্রল করে পড়তে পারবেন এবং **q** চাপলে প্রম্পট ফেরত আসবে।



## নেভিগেশন(Navigation)

আমরা সবাই ফাইল ম্যানেজারের সাথে পরিচিত। তবে ফোল্ডারে ফোল্ডারে সাজিয়ে রাখা আমাদের ফাইলগুলো। যেগুলোকে আমরা ফাইল ম্যানেজারের মাধ্যমে নিয়ন্ত্রণ করি। লক্ষ্য করলেই বুঝবেন অতি সামান্য কাজেও আমাদের ফাইলপতর নাড়াচাড়া করতে হয়। অর্থাৎ, ফোল্ডারগুলোতে ঢুকে ফাইল নিয়ে নাড়াচাড়া করাটা আসলে খুবই গুরুত্বপূর্ণ ব্যাপার। আর এটা কমান্ডলাইনে করতে গেলে আপনার জানতে হবে কীকরে ফোল্ডারের অরণ্যে বিচরণ করতে হয়।

## রুট(Root) ডিরেক্টরি(Directory)

লিনাক্সের পরিভাষায় ফোল্ডারগুলোকে সাধারণত ডিরেক্টরি(Directory) বলা হয়। লিনাক্সের যাবতীয় ফোল্ডারগুলো একটি ফোল্ডারের ভেতর থাকে একে রুট ফোল্ডার বা রুট ডিরেক্টরি বলে।

## পাথ(path)

কোনো ডিরেক্টরি বা ফাইলের অবস্থানকে তার পাথ(path) বলা হয়। পাথ দু'রকমের হয়:

- **Absolute path** বা পরম অবস্থান: রুট(root) ফোল্ডারের সাপেক্ষে পাথ হলো এ্যাবসলিউট পাথ।
- **Relative path** বা আপেক্ষিক অবস্থান: অন্য কোন পাথের সাপেক্ষে পাথ হল রিলেটিভ পাথ।

উদাহরণে পরিষ্কার হওয়ার চেষ্টা করা যাক:

```
/home/me/e_books/IT/head_first_python.pdf
```

```
~/e_books/IT/head_first_python.pdf
```

উভয়ই আসলে একই ফাইলকে নির্দেশ করে। প্রথমটি ফাইলটির এ্যাবসলিউট এবং দ্বিতীয়টি রিলেটিভ পাথ।

প্রথমটির একদম শুরুতে আছে '/' চিহ্নটি। এই চিহ্নটি শুরুতে দিলে রুট ফোল্ডার বোঝায়। তারপর পরপর ফোল্ডারগুলোর নাম আছে। এই ফোল্ডারগুলোর নাম আবার '/' চিহ্ন দিয়ে আলাদা করা। সবশেষে ফাইলটির নাম। এটি ফাইলটির এ্যাবসলিউট পাথ কেননা এতে একদম রুট ডিরেক্টরি থেকে শুরু করে কোন কোন ফোল্ডারের ভেতর ঢুকলে ফাইলটি পাওয়া যাবে তা দেখানো হচ্ছে। দ্বিতীয়টির শুরুতে আছে '~' চিহ্নটি। এটা কোনো নির্দিষ্ট ফোল্ডার না। যখন যে ইউজার ব্যবহার করছে তার হোম ওটা। আমার ইউজারনেম **me**। সুতরাং আমার হোমফোল্ডার `/home/me/`। আমার জন্য `~` আর `/home/me/` একই ব্যাপার। কারো ইউজারনেম **tuxboy** হলে তারক্ষেত্রে `~` হবে `/home/tuxboy/`। এই রিলেটিভ পাথটি হোম ডিরেক্টরির সাথে সাপেক্ষে `head_first_python.pdf` ফাইলটির অবস্থান নির্দেশ করে।

ফাইল বা ডিরেক্টরির পাথের ব্যাপারে দুটো জিনিস মনে রাখতে হবে।

1. পাথনেম কেস সেনসিটিভ। অর্থাৎ, `e_books` কে `E_BOOKS` বা `IT` কে `it` লিখলে হবে না। যেমন যেমন ঠিক

তেমন তেমনই লিখতে হবে।

2. পাথনেমে স্বাভাবিকভাবে কোনো স্পেস হয় না। অর্থাৎ Untitled Folder কে টার্মিনালে Untitled Folder লিখলে এরর দেখাবে। প্রতিটি স্পেসের আগে ব্যাকস্ল্যাশ অর্থাৎ '\' চিহ্ন দিতে হবে। অর্থাৎ Untitled Folder হবে Untitled\ Folder। স্পেসের আগে ব্যাকস্ল্যাশ না দিলে শেল Untitled এবং Folder দুটিকে আলাদা আলাদা দুটি পাথ ভাবে।

## ডিরেক্টরিতে ঘোরাফেরা

আমরা যখন ফাইল ম্যানেজার ব্যবহার করি, লক্ষ্য করবেন দুটো জিনিস আপনি সহজে জানতে পারেন। আপনি এখন কোথায় আছেন এবং আপনার সামনে অর্থাৎ এই ফোল্ডারে কি কি আছে। এদুটি জানার জন্য কমান্ড দুটি হল pwd ও ls।

```
me@howtocode-pc:~/Music$ pwd
/home/me/Music
me@howtocode-pc:~/Music$ ls
Bob Dylan                Ringtones  জাতিস্মর
Music                    Veer        রতন দা
Pete Seeger - The Essential Pete Seeger (2005) জন এর গান রবীন্দ্রসমীত
```

প্রথম উদাহরণটি **pwd** কমান্ডের। কমান্ডের পুরো নাম **print working directory**। অর্থাৎ, যে ডিরেক্টরিতে কাজ করছি তার নাম জানাও। **pwd** বর্তমান ডিরেক্টরির এ্যাবসলিউট পাথ বলে দেয়।

দ্বিতীয় উদাহরণটি **ls** কমান্ডের। এটি বর্তমান ডিরেক্টরির মধ্যের অন্যান্য ফাইল ও ফোল্ডারের নাম লিস্ট করে দেয়। এই কমান্ডের আরো চমৎকার কার্যকারিতা আছে যা পরে আলোচিত হবে।

এবার ভাবা যাক অন্য ডিরেক্টরিতে কীভাবে যাবো। এটার জন্য আমাদের দরকার হয় **cd** কমান্ডের। এই কমান্ডের আর্গুমেন্ট হিসেবে আপনি যে পাথনেম দেবেন সেখানে ঢুকে পড়বেন।

```

me@howtocode-pc:~/Music$ pwd
/home/me/Music
me@howtocode-pc:~/Music$ ls
Bob Dylan                                Ringtones  জাতিস্মর
Music                                    Veer        রতন দা
Pete Seeger - The Essential Pete Seeger (2005)  জন এর গান  রবীন্দ্রসমীত
me@howtocode-pc:~/Music$ cd Ringtones/
me@howtocode-pc:~/Music/Ringtones$ pwd
/home/me/Music/Ringtones
me@howtocode-pc:~/Music/Ringtones$ ls
[4]_Its_my_life.ogg                      Ji-Hae Park Fast 2.ogg
Amar mote by Lopamudra.ogg               Kya ye Mohabbat.ogg
Amar mote by Rupankar.ogg                 Lucky By Jason Marz for jol.ogg
Amar mote Duet.ogg                       Lucky By Jason Marz for me.ogg
Amio adhar by James.ogg                   Majhe Majhe Tobo Dekha Pai.ogg
Basuria.ogg                              Maula by Ali Azmat.ogg
Beethoven_Piano-sonata-17-Storm.ogg       MOZART~1.ogg
Bhoy Dekhas Na.ogg                       Nadan Parindey From Rockstar.ogg
Droplet.ogg                              Noyono Tomare.ogg
Excuse_boss.ogg                           Shadow Fight 1.ogg
Excuse Me Darling.ogg                     Shadow Fight 2.ogg
Grieg _The-dance-of-Anitra.ogg             sherlocker_xmeo4wao.ogg
Hariye Giyechi by Arnob.ogg                sounds-812-droplet.ogg
Irin Adler Exotic Moan.ogg                  Tumi akhono.ogg
I won't give up By Jason Mraz.ogg           Tumse Hi by Mohit chauhan.ogg
Jahazi By Shironamhin.ogg                  We shall overcome kao kao.ogg
Jani Dekha Hobe.ogg                       We shall overcome.ogg
Ji-Hae Park Fast 1.ogg
me@howtocode-pc:~/Music/Ringtones$ cd ../
me@howtocode-pc:~/Music$ cd ~
me@howtocode-pc:~$ cd /home/me/Music/Music/1.Bangla/
me@howtocode-pc:~/Music/Music/1.Bangla$ pwd
/home/me/Music/Music/1.Bangla

```

উদাহরণের শুরুতে আমরা **pwd** দিয়ে দেখে নিয়েছি যে আমরা `/home/me/Music/` ফোল্ডারে আছি। এবং তারপর **ls** কমান্ডটি দিয়ে দেখেছি ওই ফোল্ডারে আর কি কি আছে। যেহেতু Ringtones নামের ফোল্ডারটি এই(`/home/me/Music/`) ফোল্ডারের মধ্যেই আছে তাই এই ফোল্ডার থেকে Ringtones ফোল্ডারে যেতে আমরা সহজ রিলেটিভ পাথ ব্যবহার করেছি। `cd Ringtones/` কমান্ড দিয়ে আমরা Ringtones ফোল্ডারে ঢুকেছি। তারপর **pwd** ও **ls** কমান্ড দিয়ে আমাদের অবস্থান ও Ringtone ফোল্ডার এর কন্টেন্ট দেখেছি। আবার আমরা `/home/me/Music/` অর্থাৎ ঠিক উপরের ফোল্ডারে যেতে ব্যবহার করেছি `cd ../` এটা ঠিক ফাইল ব্রাউজারে আপ বাটন চাপার মত কাজ করে। এরপর আমরা হোমে ফিরে গেছি হোমের চিহ্ন '~' ব্যবহার করে এই কমান্ড দিয়ে `cd ~`। এবার আমরা এবসলিউট পাথ ব্যবহার করে `/home/me/Music/Music/1.Bangla/` ফোল্ডারে এসেছি এই কমান্ড দিয়ে: `cd /home/me/Music/Music/1.Bangla/`। **bash** এর একটা মজার ফিচার হল ট্যাব কম্প্রিটেশন। অর্থাৎ কমান্ড এর বিভিন্ন অংশ, ফাইলপাথ খানিকটা লিখে ট্যাব চাপলে বাকিটুকু সে নিজেই লিখে দেয়। অতএব আপনার দীর্ঘলাইন লিখতে হবে না। খানিকটা লিখে ট্যাব চেপে আবার প্রয়োজনে তারপরের খানিকটা লিখে ট্যাব চেপে চেপে আপনি সহজে সঠিক পাথে ঢুকতে পারবেন।



## আরো একটু ls

ls সম্ভবত সবচেয়ে বেশি ব্যবহৃত কমান্ড। ফাইলসিস্টেমে এটিই আপনার চোখের কাজ করে। আমরা আগে ls এর সাধারণ ব্যবহার দেখেছি এবার একটু গভীরে যাবো। বর্তমানে আমরা যে ডিরেক্টরিতে আছি তার কন্টেন্ট এর লিস্ট শুধু `ls` কমান্ড দিয়েই দেখতে পারি:

```
me@howtocode-pc:~$ ls
Desktop  Documents  Music  Pictures  Public  Templates  Videos
```

কিন্তু এখনি হয়ত আমার প্রয়োজন ছিল অন্যকোনো ডিরেক্টরি, মনে করেন `/usr/` ডিরেক্টরির কন্টেন্ট জানা। আমরা কি `cd` কমান্ডের মাধ্যমে ঐ ডিরেক্টরিতে গিয়ে তারপর ফাইলগুলোর লিস্ট করবো `ls` দিয়ে? তার প্রয়োজন হবে না। আমরা আমাদের বর্তমান ডিরেক্টরিতে বসেই `/usr/` ডিরেক্টরির কন্টেন্ট জানতে পারবো:

```
me@howtocode-pc:~$ ls /usr/
bin  games  include  lib  lib32  local  sbin  share  src
```

তারমানে যেকোনো ডিরেক্টরির কন্টেন্ট জানতে `ls` এর আর্গুমেন্ট হিসেবে সেই ডিরেক্টরির নাম দিলেই হবে। এমনকি আমরা চাইলে একটি না, একাধিক ডিরেক্টরির কন্টেন্ট এভাবে দেখতে পারি `ls` এর পর স্পেস দিয়ে আলাদা করে ডিরেক্টরিগুলোর নাম লিখে:

```
me@howtocode-pc:~$ ls ~ /usr/
/home/me:
Desktop  Documents  Music  Pictures  Public  Templates  Videos

/usr/:
bin  games  include  lib  lib32  local  sbin  share  src
```

আমরা `ls` এর কিছু অপশন ব্যবহার করে তার কার্যকারিতা বাড়াতে পারি। আমরা কন্টেন্টের আরো বিস্তারিত তথ্য জানতে পারি `ls` এর সাথে `-l` অপশনটি ব্যবহার করে। মনে করুন `/usr/` ডিরেক্টরির কন্টেন্ট সম্পর্কে আমরা ডিটেইল জানতে চাই। তাহলে লিখবো:

```
me@howtocode-pc:~$ ls -l /usr/
total 172
drwxr-xr-x  2 root root 69632 Aug 29 10:12 bin
drwxr-xr-x  2 root root  4096 Jul  9 15:13 games
drwxr-xr-x 68 root root 20480 Aug 22 14:32 include
drwxr-xr-x 187 root root 36864 Aug 29 10:01 lib
drwxr-xr-x  3 root root  4096 Jul 27 19:37 lib32
drwxr-xr-x 10 root root  4096 Apr 17 02:45 local
drwxr-xr-x  2 root root 12288 Aug 29 10:01 sbin
drwxr-xr-x 333 root root 12288 Aug 29 10:01 share
drwxr-xr-x 10 root root  4096 Aug 28 19:53 src
```

আবার আমরা যদি চাই ফাইল মোডিফিকেশনের ডেট অনুযায়ী এই লিস্টটি সাজানো হবে তাহলে **-t** অপশনটি ব্যবহার করবো। আগেই জেনেছি একাধিক অপশন একসাথে লেখা যায়। অর্থাৎ **-l** ও **-t** কে একসাথে **-lt** লিখবো:

```
me@howtocode-pc:~$ ls -lt /usr/
total 172
drwxr-xr-x  2 root root 69632 Aug 29 10:12 bin
drwxr-xr-x 187 root root 36864 Aug 29 10:01 lib
drwxr-xr-x  2 root root 12288 Aug 29 10:01 sbin
drwxr-xr-x 333 root root 12288 Aug 29 10:01 share
drwxr-xr-x 10 root root  4096 Aug 28 19:53 src
drwxr-xr-x 68 root root 20480 Aug 22 14:32 include
drwxr-xr-x  3 root root  4096 Jul 27 19:37 lib32
drwxr-xr-x  2 root root  4096 Jul  9 15:13 games
drwxr-xr-x 10 root root  4096 Apr 17 02:45 local
```

লক্ষ্য করুন, ফাইলগুলো তারিখ ও সময় অনুযায়ী সাজানো। এখানে সর্বশেষ মোডিফাইড ফাইল বা ডিরেক্টরি সবার প্রথমে এবং এভাবেই সাজানো। আমরা যদি চাই এটা এর বীপরীতক্রমে সাজাতে অর্থাৎ সবচেয়ে অতীতে মোডিফাইড ফাইল বা ডিরেক্টরি সবার উপরে থাকবে তাহলে **--reverse** লং অপশনটি(long option) ব্যবহার করতে পারি:

```
me@howtocode-pc:~$ ls -lt --reverse /usr/
total 172
drwxr-xr-x 10 root root  4096 Apr 17 02:45 local
drwxr-xr-x  2 root root  4096 Jul  9 15:13 games
drwxr-xr-x  3 root root  4096 Jul 27 19:37 lib32
drwxr-xr-x 68 root root 20480 Aug 22 14:32 include
drwxr-xr-x 10 root root  4096 Aug 28 19:53 src
drwxr-xr-x 333 root root 12288 Aug 29 10:01 share
drwxr-xr-x  2 root root 12288 Aug 29 10:01 sbin
drwxr-xr-x 187 root root 36864 Aug 29 10:01 lib
drwxr-xr-x  2 root root 69632 Aug 29 10:12 bin
```

এবার আসুন, **ls** এর কিছু অপশন জেনে নেয়া যাক:

অপশন	লং অপশন	বর্ণনা
-a	--all	লিস্টে সকল ফাইল ও ডিরেক্টরি দেখাবে। এমনকি '.' দিয়ে শুরু হওয়াগুলোও যেগুলো সাধারণত লুকোনো থাকে।
-A	--almost-all	এখানেও সকল ডিরেক্টরি ও ফাইল দেখাবে কিন্তু বর্তমান ডিরেক্টরির চিহ্ন './' এবং প্যারেন্ট ডিরেক্টরির চিহ্ন '../' দেখাবে না।
-d	--directory	এই অপশন ব্যবহার করলে কোনো ডিরেক্টরির কন্টেন্ট দেখাবে না বরং সেই ডিরেক্টরিকে লিস্ট আইটেম হিসেবে দেখাবে। -l অপশনের সাথে ব্যবহার করে ওই ডিরেক্টরি সম্পর্কিত তথ্য জানা যায়।
-F	--classify	এই অপশন ব্যবহার করলে লিস্টের ডিরেক্টরিগুলোর পিছনে একটি অতিরিক্ত '/' চিহ্ন ব্যবহার করবে যেন তাদের ফাইল থেকে আলাদাভাবে চিহ্নিত করা যায়।
-h	--human-readable	-l অপশন ব্যবহার করলে যে ডিটেইল লিস্ট আসে তাতে ফাইলের সাইজ বাইটে দেখায়। যা মানুষের পড়ার জন্য বেশ ঝামেলার। এই অপশনের মাধ্যমে সাইজ এমনভাবে দেখানো হয় যেভাবে আমরা পড়ে অভ্যস্ত।
-l		বহুল ব্যবহৃত ডিটেইলসহ লং ফরম্যাট।
-r	--reverse	বীপরীতক্রমে লিস্ট দেখাবে।
-s		ফাইলের সাইজ অনুযায়ী সাজাবে।
-t		মোডিফিকেশন এর সময় অনুযায়ী সাজাবে।

## লং অপশনের কাটাচ্ছেড়া

লং অপশনের ডিটেইল লিস্ট থেকে দেখানো একটা লাইন আমরা বিবেচনায় নিই:

```
-rwxrwxrwx 1 me root      2498111 Jul 10 11:31 01. Blowin' In The Wind.ogg
```

এইখানে আমরা এই অংশগুলো দেখতে পাই:

- ফাইল পারমিশন(**File permission**): প্রথমে যে **-rwxrwxrwx** অংশটা দেখাচ্ছে ওটা ফাইল পারমিশন নির্দেশ করে। প্রথমে - থাকলে সেটা সাধারণ ফাইল, **d** থাকলে ডিরেক্টরি। তারপরের অংশগুলো নির্ধারন করে কে বা কারা ফাইলে কোনপর্যায়ের এক্সেস পাবে। আমরা পরে এইনিয়ে বিস্তারিত আলোচনা করবো।
- হার্ড লিঙ্ক সংখ্যা: এখানে '1'। ফাইলটির কতগুলো হার্ড লিঙ্ক আছে। লিঙ্ক নিয়ে আমরা পরে বিস্তারিত জানবো।
- ফাইলটির মালিকের ইউজারনেম: এখানে **me**। যে এই ফাইলটির মালিক বা ক্রিয়েটর তার ইউজারনেম এখানে থাকবে।
- ফাইলটির মালিক গ্রুপের নাম: ফাইলটি যে গ্রুপের মালিকানায়, তার নাম। এখানে **root**
- ফাইলটির আয়তন বা সাইজ: এখানে বাইটের এককে ফাইল সাইজ দেখানো হয়। আমাদের এই ফাইলটির সাইজ 32059 বাইট।
- ফাইল মোডিফিকেশন টাইমস্ট্যাম্প: ফাইলটি শেষ কবে এবং কখন মোডিফাই করা হয়েছে তাই এখানে জানানো হয়। আমাদের ফাইলটির ক্ষেত্রে যা ছিল **Jul 10 11:31**।

- ফাইলের নাম: সবশেষে ফাইলের নাম । এক্ষেত্রে **01. Blowin' In The Wind.ogg** ।



# ফাইল

লিনাক্সের দুনিয়ায় একটি কথা খুব জনপ্রিয়, "Everything is a file." অর্থাৎ সবকিছুই ফাইল। ফাইল নিয়েই যত কাজ। এই অংশে আমরা ফাইল সম্পর্কিত কিছু কমান্ড নিয়ে কথা বলবো।

## ফাইল চেনা

একটা ফাইল সম্পর্কে মোটামুটি একটা ধারণা আসবে **file** কমান্ডটি দিয়ে। এই কমান্ডটির আর্গুমেন্ট হিসেবে ফাইলের নাম দিলে সে বের করে দেবে এর সম্পর্কিত কিছু তথ্য।

```
me@howtocode-pc:~/Music/Bob Dylan$ file 01.\ Blowin'\ In\ The\ Wind.ogg
01. Blowin' In The Wind.ogg: Ogg data, Vorbis audio, stereo, 44100 Hz, ~112000 bps
```

উপরের উদাহরণে আমরা একটি ওজিজি অডিও ফাইলের সম্পর্কে জানতে চেয়েছিলাম **file** কমান্ডটির সাহায্যে। এটি আমাদের বেশকিছু তথ্য জানিয়েছে।

## টেক্সট ফাইল পড়া

আমরা **less** কমান্ডটি ব্যবহার করি টেক্সট(text) ফাইল পড়তে। লিনাক্স সিস্টেমে প্রচুর প্রোগ্রাম সরাসরি টেক্সট ফাইল ব্যবহার করে। **less** এইসব ফাইল পড়ার বেশ একটা দ্রুত ও সহজ উপায় করে দেয়। একটা মজার জিনিস বলে রাখি। ইউনিক্স সিস্টেমে একইরকম একটি প্রোগ্রাম ছিল more নামে। সেটির ফ্রী বা লিব্রে ভার্সন হল less।

## টেক্সট কি?

এখন আমাদের একটু জানা উচিত টেক্সট ফাইল কোনগুলোকে বলবো। কম্পিউটার বিভিন্নরকম তথ্য আমাদের সামনে বিভিন্নভাবে উপস্থাপন করে। এইসব তথ্য উপস্থাপনের ক্ষেত্রে দুটো বিষয় যুক্ত। ১) কি ধরনের তথ্য এবং ২) কিরকম সংখ্যায় তার প্রকাশ। কম্পিউটার শেষেমেষ সংখ্যা ছাড়া আর কিছুই বোঝে না।

কোনো কোনো ক্ষেত্রে এই প্রকাশপদ্ধতি খুব জটিল যেমন কমপ্রেসড ভিডিও ফাইল, অন্যদিকে কিছু কিছু বেশ সহজ। একটা খুব প্রথমদিকের ও সহজ উপায় হল অ্যাসকি টেক্সট(ASCII - American Standard Code for Information Interchange)। এটা প্রথমে টেলিটাইপ মেশিনগুলোতে ব্যবহৃত হয়েছিল কীবোর্ডের ক্যারেটরগুলোর সংখ্যাতাত্ত্বিক অবস্থান দিতে। এটা খুবই সহজ পদ্ধতি এবং খুব কম জায়গা নেয়। একটি অক্ষরের জন্য এক বাইট জায়গা প্রয়োজন হয়। এক্ষেত্রে বলা ভালো, ওয়ার্ড প্রসেসর দিয়ে তৈরী ডকুমেন্ট কিন্তু টেক্সট ফাইল নয়। এতে অনেক ননক্যারেটর তথ্য থাকে ফরম্যাটিং এর জন্যে।

## less এর ব্যবহার

**less** কমান্ডটি **file** কমান্ডটির মতই ব্যবহার করতে হয়। অর্থাৎ কমান্ডের আর্গুমেন্ট হিসেবে ফাইলনেম দিতে হয়। যেমন: `less /etc/passwd`। কমান্ডটি দিলে ফাইলটি দেখাবে। আপনি উপরে ও নীচে স্ক্রল করে পড়তে পারবেন এবং **q** চেপে এটা বন্ধ করতে পারবেন।

আমরা **less** এর গুরুত্বপূর্ণ কমান্ডগুলোতে চোখ বুলিয়ে নিই:

কমান্ড	কাজ
Page Up or b	এক পেজ উপরে স্ক্রল করবে
Page Down or space	এক পেজ নীচে স্ক্রল করবে
Up Arrow	এক লাইন উপরে স্ক্রল করবে
Down Arrow	এক লাইন নীচে স্ক্রল করবে
G	ফাইলের একদম শেষে যাবে
1G or g	ফাইলের একদম শুরুতে যাবে
/characters	/ চিহ্নের সাথে যা লেখা হবে সেটা খুঁজবে এবং প্রথম যেখানে পাওয়া গেছে দেখাবে
n	কোনো কিছু সার্চ করা হলে সেটা আর পরে কোথায় আছে দেখাবে
h	হেল্প স্ক্রীন দেখাবে
q	বন্ধ করবে

# লিঙ্ক(Link)

লিনাক্স সিস্টেমে সাধারণ ফাইল ও ডিরেক্টরির বাইরেও একটা বস্তু আছে তার নাম লিঙ্ক। আপনি হয়ত কোনো ওয়েবসাইটে কোথাও ক্লিক করলে অন্য পেজে চলে যেতে দেখেছেন। এখানে ব্যাপারটা প্রায় সেরকম। লিঙ্ক আবার দূরকন্মের। সিমবোলিক লিঙ্ক(symbolic link or soft link or sym-link) এবং হার্ড লিঙ্ক(Hard link)।

## সিমবোলিক লিঙ্ক

আপনি হয়ত কখনোসখনো এমনকি কোনো কিছু দেখে থাকবেন **ls -l** কমান্ডের আউটপুটে:

```
lrwxrwxrwx 1 root root 20 Jul 9 03:51 libxtables.so.10 -> libxtables.so.10.0.0
```

এর শুরুতে 'l' নির্দেশ করে যে এটি একটি সিমবোলিক লিঙ্ক। সিমবোলিক লিঙ্কের সুবিধা হল এটি একটি ফাইলে বিভিন্ন নামে প্রকাশের সুযোগ দেয়। এরর প্রয়োজনীয়তা কি? আসুন একটু ভেবে দেখা যাক। লিনাক্স একটি ফ্রী(মুক্ত) সফটওয়্যার হওয়ায়, প্রোপাইটরি সফটওয়্যার প্রস্তুতকারকদের মত নিজেদের রিসোর্স আবদ্ধ রাখেনা। অধিকাংশ রিসোর্স বিশেষ করে লাইব্রেরি ফাইলগুলো একাধিক প্রোগ্রাম দ্বারা ব্যবহৃত হয়। মনে করেন, **libfoo** নামের একটা লাইব্রেরি ফাইল একটা প্রোগ্রাম ব্যবহার করে। কিন্তু এই libfoo হয়ত প্রতিনিয়ত আপডেট নেয়। এবং স্বাভাবিকভাবেই প্রত্যেক ডারশনের নম্বর ফাইলটির সাথে যুক্ত হয়। এতে সমস্যাও সৃষ্টি হয় একটি। প্রত্যেকবার libfoo এর নতুন ভার্সন আসার সাথে সাথে আমাদের প্রত্যেকটা প্রোগ্রাম খুঁজে বের করতে হবে যারা এই লাইব্রেরি ব্যবহার করে আর তাদের এমনভাবে পরিবর্তন যেন তারা নতুন ভার্সনটিকে ব্যবহার করতে পারে। বাস্তবজীবনে যা খুবই পরিশ্রমসাধ্য।

সিমবোলিক লিঙ্ক দিয়ে সহজেই এই সমস্যার সমাধান করা যায়। মনে করি আমরা **libfoo-2.6** ইন্সটল করেছি এবং একটা সিমবোলিক লিঙ্ক তৈরী করেছি শুধুমাত্র **libfoo** নামে। যে প্রোগ্রামাররা libfoo কে তাদের প্রোগ্রামে ব্যবহার করবে তারা ওই লিঙ্কটি ব্যবহার করবে। ফলে আমরা পরে libfoo-2.7 ইন্সটল করে যখন একই নামে অর্থাৎ **libfoo** তে লিঙ্ক করবো তাকে তার প্রোগ্রামে পরিবর্তন আনতে হবে না।

সিমবোলিক লিঙ্ক হার্ড লিঙ্কের চেয়ে অনেক আধুনিক প্রযুক্তি। হার্ডলিঙ্ক তার নিজের ডাইভের বাইরে লিঙ্ক তৈরী করতে পারে না এবং ডিরেক্টরির হার্ড লিঙ্ক হয় না। সিমবোলিক লিঙ্কের এই সীমাবদ্ধতাগুলো নেই। সিমবোলিক লিঙ্ক আর তার অরিজিনাল ফাইলটির ব্যবহারযোগ্যতায় কোনো পার্থক্য নেই। তবে লিঙ্কটি ডিলিট করলে আপনি শুধু লিঙ্কটিই ডিলিট করবেন, অরিজিনাল ফাইলটি নয় এবং অরিজিনাল ফাইলটি ডিলিট করলে লিঙ্কটি ব্রোকেন লিঙ্ক হিসেবে থাকবে এবং কোনো কিছু দিকেই পয়েন্ট করবে না।

## হার্ড লিঙ্ক

হার্ড লিঙ্ক ইউনিক্স সিস্টেমের আসল লিঙ্কিং পদ্ধতি। প্রত্যেক ফাইলের অন্তত একটি হার্ড লিঙ্ক থাকে যা তাকে তার নামটি দেয়। এবং হার্ড লিঙ্কে তার ফাইলের সাথে আলাদা করা যায় না। আপনি যদি একটি হার্ড লিঙ্ক ডিলিট করেন যদি সেই ফাইলটির আর কোনো হার্ড লিঙ্ক না থাকে তাহলে সেটি ডিলিট হয়ে যাবে। আর আরো হার্ড লিঙ্ক থাকলে

যতক্ষণ না সবগুলো লিঙ্ক ডিলিট করছেন ফাইলটি ডিলিট হবে না। আগেই বলেছি, হার্ড লিঙ্কের দুটি সীমাবদ্ধতা আছে। এটি নিজের ফাইলসিস্টেম এর বাইরে, সহজ কথায় নিজের ড্রাইভের বাইরে লিঙ্ক করতে পারে না এবং ডিরেক্টরির হার্ড লিঙ্ক হয় না।

## লিনাক্স ফাইলসিস্টেম

গত কয়েকটি লেসনে আমরা জেনেছি **cd** কমান্ড দিয়ে কীভাবে ডিরেক্টরিতে ঢুকতে হয়, **ls** কমান্ড দিয়ে কীভাবে ডিরেক্টরির কন্টেন্ট এর লিস্ট দেখতে হয়(কাজের ক্ষেত্রে **ls -l** বেশি সুবিধাজনক), **file** কমান্ড দিয়ে কীভাবে ফাইল সম্পর্কে গুরুত্বপূর্ণ তথ্য জানতে হয় এবং **less** দিয়ে কীভাবে টেক্সট ফাইলের কন্টেন্ট দেখতে হয়। আপনার উচিত এখন বিভিন্ন ডিরেক্টরিতে ঢোকা। ইন্টারেস্টিং ফাইল পেলে তার সম্পর্কে জানা বা **less** দিয়ে পড়ার চেষ্টা করা। আপনি নির্ভয়ে এটি করতে পারেন। নিশ্চিত থাকতে পারেন কোনো এলিয়েন এসে এই কারনে আপনার পোষা বিড়ালকে হত্যা করবে না! আপনার আগ্রহে উস্কানি দিতে আসুন কিছু গুরুত্বপূর্ণ ডিরেক্টরি সম্পর্কে জানি:

ডিরেক্টরি	মন্তব্য
<b>/</b>	রুট ডিরেক্টরি। এখান থেকেই সবকিছুর শুরু।
<b>/bin</b>	সিস্টেম চালু হতে ও সচল রাখতে দরকারি বাইনারিতে কম্পাইল করা প্রোগ্রামগুলো এখানে থাকে।
<b>/boot</b>	লিনাক্স কার্নেল( <b>/boot/vmlinuz</b> ), সিস্টেম চালু হতে দরকারি কিছু ড্রাইভার সমেত RAM disk image ও বুট লোডার।
<b>/dev</b>	এই ডিরেক্টরিতে ডিভাইস নোডগুলো (device nodes) থাকে। অর্থাৎ, সিস্টেমের জানাশোনা সব ডিভাইসের একটি লিস্ট।
<b>/etc</b>	সিস্টেমের সকল কনফিগারেশন ফাইল ও কিছু শেলস্ক্রিপ্ট এখানে থাকে সিস্টেম চালু হওয়ার সময় লোড হয়। এই ডিরেক্টরির সবকিছুই মানুষ পড়তে পারে। যেমন <b>crontab</b> ফাইলে কিছু অটোমেটেড কাজের নির্দেশনা থাকে। <b>fstab</b> ফাইলে মাউন্টেড ডিভাইস সম্পর্কিত তথ্য থাকে। <b>passwd</b> ফাইলে ইউজারদের একটা লিস্ট।
<b>/home</b>	প্রত্যেক ইউজারের জন্য একটি করে ফোল্ডার থাকে এইখানে। সাধারণ ইউজার শুধু তার নিজের ডিরেক্টরিতে কাজ করতে পারে।
<b>/lib</b>	সিস্টেমের ব্যবহৃত লাইব্রেরি ফাইলগুলো এখানে থাকে।
<b>/lost+found</b>	লিনাক্স ফাইলসিস্টেম( যেমন ext3, ext4) এ ফরম্যাট করা সব ড্রাইভেই এই ফোল্ডার থাকে। ডাটা করাপশন থেকে রিকভার করা ফাইলগুলো এখানে পাওয়া যেতে পারে। গুরুতর কোনো সমস্যা না থাকলে এই ডিরেক্টরি ফাঁকাই থাকে।
<b>/media</b>	আধুনিক লিনাক্স সিস্টেমে ইউএসবি ড্রাইভ, সিডি বা ডিভিডি গুলো অটোমেটিক মাউন্ট হয় ও এই ডিরেক্টরিতে একটি সাবডিরেক্টরি হিসেবে তাদের পাওয়া যায়।
<b>/mnt</b>	যেসব ডিভাইসকে ম্যানুয়ালি মাউন্ট করা হয়, তাদের এখানে পাওয়া যায়।
<b>/opt</b>	এখানে অপশনাল সফটওয়্যার, মূলত কমার্শিয়াল সফটওয়্যারগুলো ইন্সটল হয়।
<b>/proc</b>	এই ডিরেক্টরিটি সরাসরি কার্নেল নিয়ন্ত্রণ করে। কার্নেল সম্পর্কিত বিভিন্ন তথ্য এখানে পাওয়া যাবে যা পড়ার যোগ্য।
<b>/root</b>	কম্পিউটারের রুট এ্যাকাউন্টের হোম ডিরেক্টরি এটা।
<b>/sbin</b>	এখানে সিস্টেম বাইনারি প্রোগ্রামগুলো থাকে। যেগুলো মূলত সুপারইউজারের ব্যবহারের জন্য রাখা হয়।
<b>/tmp</b>	বিভিন্ন প্রোগ্রাম তার প্রয়োজনমত টেম্পোরারি ফাইল তৈরি করে এখানে। প্রত্যেক রিবুট বা

<b>/tmp</b>	রিস্টার্টে এই ডিরেক্টরির সবকিছু মুছে যায়।
<b>/usr</b>	সাধারণ ব্যবহারকারীর সকল সফটওয়্যার ও তার সাপোর্ট ফাইলগুলো এখানে থাকে।
<b>/usr/bin</b>	ইন্সটল করা সফটওয়্যারের এক্সিকিউটেবল বাইনারি এখানে থাকে।
<b>/usr/lib</b>	/usr/bin এর প্রোগ্রামগুলোর লাইব্রেরি ফাইল এখানে থাকে।
<b>/usr/local</b>	ডিস্ট্রিবিউশন যে সফটওয়্যারগুলো সরবরাহ করে না যেমন সোর্স থেকে নিজেরা কম্পাইল করা প্রোগ্রামগুলো এখানে থাকে।
<b>/usr/sbin</b>	কিছু এডমিনিস্ট্রেশন প্রোগ্রাম এখানে থাকে।
<b>/usr/share</b>	/usr/bin এর প্রোগ্রামগুলোর ব্যবহৃত সকল শেয়ারড তথ্য যেমন কনফিগারেশন ফাইল, আইকন, স্ক্রীন ব্যাকগ্রাউন্ড, অডিও ফাইল সব এখানে থাকে।
<b>/usr/share/doc</b>	প্রোগ্রাম সম্পর্কিত ডকুমেন্টেশন ফাইলগুলো এখানে থাকে।
<b>/var</b>	ভেরিয়েবল কন্টেন্ট, যা নিয়মিত চেঞ্জ হয়, সেগুলো এখানে পাওয়া যায়। যেমন, ডাটাবেজ, স্পুল ফাইল, প্যাকেজ ম্যানেজমেন্ট আর্কাইভ।
<b>/var/log</b>	সিস্টেমের লগ ফাইলগুলো এখানে থাকে।

## অধ্যায় - দুই

# ম্যানিপুলেশন

ফাইল ও ডিরেক্টরির সাধারণ ম্যানিপুলেশন যেমন কাট, কপি, লিঙ্কিং এবং রিমুভিং বা ডিলিটেশন এই অধ্যায়ের বিষয়বস্তু।

সত্যি কথা বলতে, কখনো কখনো এসব কাজ সাধারণ গ্রাফিক্যাল ফাইল ম্যানেজারেই বেশি সুবিধাজনক। যেমন কোনো ফোল্ডার থেকে কন্টেন্ট সিলেক্ট করে টেনে অন্য ফোল্ডারে ছেড়ে দিলেই কাট হয়ে পেস্ট হয় যায়। তাহল কম্যান্ডলাইন কেন? কারন কম্যান্ডলাইন বেশি ফ্লেক্সিবল। মনে করুন আপনি একজন ওয়েব ডেভেলপার। আপনার কম্পিউটারে একটি সার্ভারও আছে। আপনি সেখানেই ভিন্ন একটি জায়গায় সাইট ডেভেলপ করেন এবং সবকিছু ঠিক থাকলে সার্ভারের সঠিক ফোল্ডারে কপি করেন। এখন যদি আপনি চান শুধু html ফাইলগুলোই কপি করবেন যেগুলো সার্ভারের ফোল্ডারে নেই বা থাকলেও তারচেয়ে আপনার ভার্সনটা নতুন। আপনি ফাইল ম্যানেজার দিয়ে খুঁটেবেছে কাজটি করতে পারেন বা আপনি করতে পারেন কম্যান্ডলাইনে, একটি কমান্ডে:

```
cp -u *.html destination_directory/
```

আসুন, শুরু করা যাক!

- **ওয়াইল্ডকার্ড:** ওয়াইল্ডকার্ডের প্রাথমিক ধারণা।
- **ফাইল ও ডিরেক্টরি তৈরি করা:** **mkdir** এবং **touch** কমান্ডের ব্যবহার।
- **ফাইল ও ডিরেক্টরি কপি করা:** **cp** কমান্ডের ব্যবহার।
- **ফাইল ও ডিরেক্টরি মুভ করা:** **mv** কমান্ডের ব্যবহার।
- **ফাইল ও ডিরেক্টরি রিমুভ করা:** **rm** কমান্ডের ব্যবহার।
- **হার্ডলিঙ্ক ও সফটলিঙ্ক তৈরি করা:** **ln** কমান্ডের ব্যবহার।
- **অনুশীলন:** পূর্ববর্তী অনুচ্ছেদগুলোতে পরিচিত কমান্ডগুলোর অনুশীলন।

## ওয়াইল্ডকার্ড(wildcards)

এই চ্যাপ্টারের শুরুতে আমরা একটা উদাহরণ দিয়েছি:

```
cp -u *.html destination_directory/
```

আসুন এটার ব্যাখ্যা করা যাক:

- **cp -u:** আমরা কপি করার কমান্ড **cp** কে **-u** অপশনসহ ব্যবহার করেছি। এই অপশনটি ব্যবহার করলে শুধু সেই ফাইলগুলো কপি হবে যেগুলো ডেসটিনেশন ডিরেক্টরিতে(অর্থাৎ যেখানে কপি করবো) নেই কিন্তু সোর্স ডিরেক্টরিতে(অর্থাৎ যেখান থেকে কপি করবো) আছে বা যে ফাইলগুলোর লেটেস্ট ভার্সন সোর্স ডিরেক্টরিতে আছে আর ডেসটিনেশন ডিরেক্টরিতে আছে পুরনো ভার্সন।
- **\*.html:** আমরা \*.html দিয়ে সকল html ফাইলগুলো সিলেক্ট করেছি। অর্থাৎ শুধুমাত্র html ফাইলগুলোই কপি করবে। '\*' চিহ্নের এই ব্যবহারকে ওয়াইল্ডকার্ড বলে। এই লেসনে আমরা ওয়াইল্ডকার্ডের ব্যবহার নিয়েই কথা বলবো।
- **destination\_directory/:** যে ডিরেক্টরিতে ফাইলগুলো কপি করা হবে।

আমরা দেখলাম ওয়াইল্ডকার্ড আমাদের একটা একটা করে ফাইলের নাম লেখার যন্ত্রণা থেকে বাঁচিয়েছে। এটি শেলের একটি ফিচার। বেশ কয়েকটি ওয়াইল্ডকার্ড আছে আমাদের পরিশ্রম কমানোর জন্য।

ওয়াইল্ডকার্ড	অর্থ
*	যেকোনো ক্যারেक्टर সিলেক্ট করবে।
?	যেকোনো একটি ক্যারেक्टर সিলেক্ট করবে।
[charecters]	সেটের অন্তর্গত যেকোনো ক্যারেक्टर সিলেক্ট করবে।
[!charecters]	সেটের অন্তর্গত নয় এমন যেকোনো ক্যারেक्टर সিলেক্ট করবে।
[:class:]	বিশেষ ক্লাসের অন্তর্গত ক্যারেक्टर সিলেক্ট করবে।

কিছু ক্যারেक्टर ক্লাস আছে যেগুলো ব্যবহার করে একই শ্রেণীর ক্যারেक्टरগুলো সহজে সিলেক্ট করা যায়:

ক্যারেक्टर ক্লাস	অর্থ
[:alnum:]	সকল অক্ষর ও সংখ্যা।
[:alpha:]	সকল অক্ষর।
[:digit:]	সকল সংখ্যা।
[:lower:]	সকল ছোটহাতের ইংরাজি অক্ষর।
[:upper:]	সকল বড়হাতের ইংরাজি অক্ষর।



আসুন ওয়াইল্ডকার্ড ব্যবহার করে তৈরি কিছু প্যাটার্ন দেখি এবং দেখি সেগুলো কি করবে:

প্যাটার্ন	অর্থ
*	সব ফাইল সিলেক্ট করবে।
g*	'g' দিয়ে শুরু হওয়া সব ফাইল সিলেক্ট করবে।
b*.txt	'b' দিয়ে শুরু হওয়া এবং '.txt' দিয়ে শেষ হওয়া সকল ফাইল সিলেক্ট করবে।
Data???	'Data' দিয়ে শুরু হওয়া ফাইল যেগুলোর পরে তিনটি ক্যারেক্টার থাকবে শুধু সেগুলো সিলেক্ট করবে।
[abc]*	'a', 'b' অথবা 'c' দিয়ে শুরু হওয়া সকল ফাইল সিলেক্ট করবে।
BACKUP.[0-9] [0-9][0-9]	'BACKUP.' দিয়ে শুরু করা যেকোনো ফাইল যার পিছনে ঠিক তিনটি সংখ্যা আছে শুধুমাত্র সেগুলো সিলেক্ট করবে।
[:upper:]*	সকল ফাইল যেগুলোর নামের শুরুতে বড়হাতের অক্ষর আছে।
[:digit:]*	সকল ফাইল যা কোনো সংখ্যা দিয়ে শুরু হয়নি।
*[[:lower:]]123]	সকল ফাইল যার শেষে হয় ছোটহাতের অক্ষর বা '1', '2' বা '3' সংখ্যাগুলো আছে।

# ফাইল ও ডিরেক্টরি তৈরী করা

## ডিরেক্টরি তৈরি করা

ডিরেক্টরি তৈরী করতে আমরা **mkdir** কমান্ডটি ব্যবহার করি। কমান্ডটির কাঠামো এরকম:

```
mkdir directory...
```

লক্ষ্য রাখবেন, পরবর্তীতে এরকম কাঠামো দেখানোর সময়ে '...' চিহ্ন থাকলে বুঝবেন যে স্পেস ব্যবহার করে ওই আর্গুমেন্টের পুনরাবৃত্তি সম্ভব। অর্থাৎ, আমরা **dir1** নামে একটি ডিরেক্টরি তৈরি করতে যেমন লিখতে পারি:

```
mkdir dir1
```

তেমনিভাবে, **dir1**, **dir2** এবং **dir3** নামের তিনটি ফোল্ডার তৈরী করতে লিখতে পারি:

```
mkdir dir1 dir2 dir3
```

## ফাইল তৈরী করা

ফাইল তৈরী করতে শেখা আলাদাভাবে জরুরী কিছু না। কেননা, যেসব প্রোগ্রাম নতুন ফাইলে সেভ রাখতে জানে তারা ফাইল তৈরী করতেও সক্ষম।

আমরা এখানে যে কমান্ডটি ফাইল তৈরী করতে ব্যবহার করবো সেটি সেইসব ফাইলের জন্য ভালো কাজ করে যেগুলো প্লেইন টেক্সট ব্যবহার করে, যেমন, txt, md, html ইত্যাদি। কমান্ডটি হল **touch** এবং এর উদ্দেশ্য আসলে ফাইলকে টুয়ে যাওয়া অর্থাৎ এর মোডিফিকেশন এর সময় বর্তমান সময় করে দেয়া। এটা করার সময় যদি ওই নামে কোনো ফাইল না থাকে তাহলে সেটি তৈরী করে দেয়। এর ব্যবহার **mkdir** এর মতই। অর্থাৎ,

```
touch filename...
```

## ফাইল ও ডিরেক্টরি কপি করা

আমরা এর পূর্ববর্তী কয়েকটি উদাহরনে **cp** কমান্ডটি দেখেছি। ফাইল ও ডিরেক্টরি কপি করতে এই কমান্ডটি ব্যবহার করা হয়। এটা ব্যবহারের দুটো উপায়। প্রথমটি:

```
cp source_item destination_item
```

এখানে **source\_item** ও **destination\_item** উভয়েই ফাইল বা ডিরেক্টরি যেকোনোকিছু হতে পারে। কোনক্ষেত্রে ফলাফল কি হবে দেখে নিই:

	সোর্স: ফাইল	সোর্স: ডিরেক্টরি
ডেস্টিনেশন: ফাইল	সোর্স ফাইলটি ডেস্টিনেশন ফাইলের জায়গায় কপি হবে। অর্থাৎ ওভাররাইড করবে। তবে ডেস্টিনেশন ফাইলের নাম ঠিক থাকবে।	এরর দেখাবে। কারন, ফাইলকে ডিরেক্টরি দিয়ে ওভাররাইড করা যায় না।
ডেস্টিনেশন: ডিরেক্টরি	সোর্স ফাইলটি ডেস্টিনেশন ডিরেক্টরির মধ্যে কপি হবে।	সোর্স ডিরেক্টরিটি ডেস্টিনেশন ডিরেক্টরির মধ্যে কপি হবে।

কপি কমান্ডের দ্বিতীয় কাঠামোটি হল:

```
cp source_item... destination_directory
```

অর্থাৎ আপনি একাধিক সোর্স ফাইল এবং ডিরেক্টরি একটি ডেস্টিনেশন ডিরেক্টরিতে কপি করতে পারবেন। মনে রাখবেন শেষ আর্গুমেন্ট টি ডিরেক্টরি হবে এবং ওইটিই ডেস্টিনেশন হবে সবসময়।

এবার আমরা **cp** কমান্ডের কিছু গুরুত্বপূর্ণ অপশনে নজর বুুলিয়ে নিই:

অপশন	লং অপশন	অর্থ
-a	-- archive	ফাইল বা ডিরেক্টরি তার সকল গুনাগুন যেমন ফাইল পার্মিশন এবং ওনারশিপ সম্পর্কিত তথ্যাবলী সহ কপি হবে।
-i	-- interactive	সোর্স ও ডেস্টিনেশনে উভয় স্থানে একই ফাইল থাকে তাহলে ওভাররাইড করা হয়। এই অপশন যুক্ত করলে ওভাররাইড করার আগে জানতে চাইবে ওভাররাইড করা হবে কিনা।
-r	-- recursive	এই অপশন(বা -a) যুক্ত করলে ডিরেক্টরি তার সকল কন্টেন্ট সহ কপি করে। নতুবা ডিরেক্টরি কপি হয়না। অর্থাৎ ডিরেক্টরি কপি করতে গেলে এই অপশনটি অবশ্যই দিতে হবে(যদি -a ব্যবহার করা না হয়।)
-u	--update	সোর্স থেকে শুধু সেই ফাইলগুলোই কপি করা হবে যেগুলো ডেস্টিনেশনে নেই বা পুরাতন সময়ের আছে।
-v	--verbose	সাধারনত, কপি করার সময় এরর না হলে কোনো তথ্যই দেখায় না। এই অপশন যোগ করলে কপি করার সম্পর্কিত তথ্যগুলি দেখাবে।

ব্যবহারিক জীবনে আমি সাধারণত সবসময় **cp -rv** ব্যবহার করি।

## ফাইল ও ডিরেক্টরি মুভ করা

ফাইল ও ডিরেক্টরি মুভ বা স্থানান্তর করতে(এবং রিনেম বা নাম পরিবর্তনের জন্যও) ব্যবহৃত কমান্ডটি হল **mv**। এর ব্যবহার **cp** কমান্ডের মতই। একটি ফাইল বা ডিরেক্টরি মুভ করতে বা রিনেম করতে প্রয়োজনীয় কমান্ডকাঠামোটি হচ্ছে:

```
mv source_item destination_item
```

এবং একাধিক ফাইল বা ডিরেক্টরি কোনো ডিরেক্টরিতে স্থানান্তর করতে:

```
mv source_item... destination_directory
```

আসুন এবার **mv** কমান্ডটির কিছু গুরুত্বপূর্ণ অপশন জেনে নেয়া যাক:

অপশন	লং অপশন	অর্থ
-i	--interactive	সোর্স থেকে মুভ করা হচ্ছে এমন কোনো ফাইল এর নাম যদি ডেস্টিনেশনে আগে থেকেই আছে এমন কোনো ফাইলের সাথে মিলে যায় তবে এই অপশনটি ব্যবহার করলে ফাইলটি ওভাররাইড করার আগে অনুমতি চাইবে। অনথ্যায় নীরবে ওভাররাইড করবে।
-u	--update	এই অপশন ব্যবহার করলে শুধুমাত্র সেইসব ফাইল সোর্স থেকে মুভ করবে যেগুলো ডেস্টিনেশনে নেই বা থাকলেও তার থেকে নতুন।
-v	--verbose	এই অপশন ব্যবহার করলে কাজের সম্পর্কে বিভিন্ন তথ্য জানাবে।

এবার কিছু উদাহরণ দেখে নেয়া যাক:

```
mv source_file destination_file
```

আমরা একটি ফাইলকে আরেকটি ফাইলে মুভ করছি। এক্ষেত্রে দুটো ব্যাপার ঘটতে পারে। `destination_file` নামে কোনো ফাইল থাকলে তার তথ্য `source_file` এর তথ্য দ্বারা ওভাররাইড হবে। যদি `destination_file` নামে কোনো ফাইল না থাকে তাহলে তৈরী হবে।

এবার `-i` অপশনযুক্ত কমান্ড দেখা যাক:

```
mv -i source_file destination_file
```

সবকিছু উপরের উদাহরণের মতই প্রায় শুধু `destination_file` নামে কোনো ফাইল আগে থেকেই থাকলে ওভাররাইড এর আগে অনুমতি চাইবে।

এবার দেখা যাক একাধিক ফাইলকে কিভাবে একটি ডিরেক্টরিতে স্থানান্তর করবো। এখানে উল্লেখ্য যে, যে ডিরেক্টরিতে মুভ করা হবে তাকে আগে থেকেই থাকতে হবে।

```
mv source_file1 source_file2 destination_directory
```

এবার ডিরেক্টরি স্থানান্তরের একটি উদাহরণ:

```
mv source_directory destination_directory
```

এক্ষেত্রে দুটো জিনিস হতে পারে। যদি destination\_directory না থাকে তবে প্রথমে destination\_directory নামে একটি ডিরেক্টরি তৈরী করবে। তারপর source\_directory থেকে সবকিছু সেখানে মুভ করবে এবং সবশেষে source\_directory ডিলিট করে দেবে।

আর যদি destination\_directory আগে থেকেই থাকে, তাহলে source\_directoryটিকে তার সব কন্টেন্টসহ destination\_directory এর মধ্যে মুভ করা হবে।

## ফাইল ও ডিরেক্টরি রিমুভ করা

কমান্ডলাইনে ব্যবহৃত সবচেয়ে ভয়ঙ্কর কমান্ডগুলোর একটি **rm** যা ফাইল ও ডিরেক্টরি রিমুভ বা ডিলিট বা মুছে ফেলতে ব্যবহৃত হয়। এর কমান্ড কাঠামো খুবই সহজ:

```
rm item...
```

**item** এর জায়গায় এক বা একাধিক ফাইল ও ডিরেক্টরি থাকতে পারে যেগুলো মুছে ফেলতে চান।

এবার এর কিছু অপশন দেখে নেয়া যাক:

অপশন	লং অপশন	অর্থ
-i	--interactive	এই অপশন ব্যবহার করলে প্রত্যেক ফাইল মুছে ফেলার আগে অনুমতি চাইবে।
-r	--recursive	এই অপশন ব্যবহার করলে ডিরেক্টরিকে তার সব কন্টেন্টসহ মুছে ফেলা হয়। আসলে ডিরেক্টরি মুছে হলে এই অপশনটির ব্যবহার জরুরী।
-f	--force	কোনো কোনো ফাইল ডিলিট করার সময় আপনাকে শেল জানাতে পারে এগুলো ডিলিট করা ঠিক হবে না তাই ডিলিট করা হচ্ছে না। এই অপশন ব্যবহার করে সমস্তকিছু ডিলিট করা সম্ভব। এই অপশনটি ব্যবহার করলে --interactive অপশন কাজ করবে না।
-v	--verbose	মুছে ফেলার প্রক্রিয়া বিস্তারিত দেখাবে।

## rm নিয়ে সতর্ক থাকবেন!

**rm** কমান্ডের ব্যবহারের সময় সতর্কতা অবলম্বন করাই বুদ্ধিমানের কাজ(আমি নিজেই একবার ভুল করে আমার সমস্ত মিডিয়া স্টোরেজ মুছে ফেলেছিলাম!)।

মনে করুন একটা ফোল্ডারে আপনার বেশকিছু ফাইল আছে। তারমধ্যে হয়ত আছে কিছু html ফাইল যা আপনার দরকার নেই। আপনি ভাবলেন ওয়াইল্ডকার্ড ব্যবহার করে এক কমান্ডে সব মুছে ফেলবেন এভাবে:

```
rm *.html
```

কমান্ডটি ঠিক থাকলে পারফেক্টলি কাজ করবে। কিন্তু যদি ভুল হয়? যদি এমন হয়:

```
rm * .html
```

অর্থাৎ \* এবং .html এর মধ্যে একটি স্পেস হলেই প্রথমে ওই ডিরেক্টরির মধ্যে থাকা সবকিছু মুছে ফেলবে এবং তারপর হয়ত একটি এরর দেখাবে যে .html বলে কোনো ফাইল পাওয়া যায়নি। টাইপ করতে ভুল হওয়া খুব স্বাভাবিক ব্যাপার।

সবচেয়ে ভালো হয়, আপনি rm দিয়ে যে কমান্ডটি দিতে চান, সেটাতে আগে একবার **rm** এর জায়গায় **ls** দিয়ে টাই করেন। তাহলে দেখতে পারবেন কি কি ডিলিট করতে চাচ্ছেন। সব ঠিক থাকলে আপনার এ্যারো কী চেপে সেই কমান্ডটি আবার আনুন এবং **ls** এর জায়গায় **rm** বসিয়ে দিন।



## হার্ডলিঙ্ক ও সফটলিঙ্ক তৈরি করা

লিঙ্ক এর জন্য ব্যবহৃত কমান্ডটি হচ্ছে **ln**। এটা অন্য কমান্ডগুলোর তুলনায় জটিলতাহীন সহজ কমান্ড। কোনো ফাইলের হার্ডলিঙ্ক তৈরী করতে:

```
ln file link
```

এবং কোনো ফাইল বা ডিরেক্টরি যেকোনোকিছু সফটলিঙ্ক তৈরি করতে:

```
ln -s item link
```

যেখানে **item** ফাইল বা ডিরেক্টরি যেকোনোকিছুই হতে পারে।

## অনুশীলন

এ পর্যায়ে আমরা এই অধ্যায়ে জানা কমান্ডগুলো একটু ব্যবহার করে দেখবো। কিন্তু আমরা চাইনা কম্পিউটারের অন্যসব তথ্য বা ফাইল স্ক্রটিগ্রস্ব করতে। তাই প্রথমে হোম ডিরেক্টরিতে playground নামে একটি ডিরেক্টরি করে নিই:

```
me@howtocode-pc:~$ cd ~
me@howtocode-pc:~$ mkdir playground
```

প্রথমে **cd ~** কমান্ড দিয়ে আমরা নিশ্চিত হয়ে নিলাম আমরা হোম ডিরেক্টরিতেই আছি এবং তারপর **mkdir playground** কমান্ড দিয়ে আমরা playground নামে একটি ডিরেক্টরি বা ফোল্ডার তৈরি করলাম। আসুন, আমরা এবার playground ডিরেক্টরিতে ঢুকি এবং dir1 ও dir2 নামে দুটো ডিরেক্টরি তৈরি করি:

```
me@howtocode-pc:~$ cd playground/
me@howtocode-pc:~/playground$ mkdir dir1 dir2
```

লক্ষ্য করুন, আমরা একটি কমান্ডেই ডিরেক্টরিদুটো তৈরি করেছি।

এবার কিছু তথ্য বা ফাইল আনা যাক। আমরা /etc ডিরেক্টরি থেকে passwd ফাইলটি বর্তমান ডিরেক্টরিতে কপি করে আনি:

```
me@howtocode-pc:~/playground$ cp /etc/passwd .
```

আমরা **cp** কমান্ডটির দুটি আর্গুমেন্ট দিয়েছি। প্রথমে সোর্স আইটেম হিসেবে **/etc/passwd** অর্থাৎ রুট ডিরেক্টরিতে যে etc ডিরেক্টরি তার passwd ফাইলটি এবং শেষে ডেস্টিনেশন হিসেবে '.' চিহ্ন যা দিয়ে বর্তমান ডিরেক্টরি বুঝায়। এবার **\*\*ls -l** দিয়ে বর্তমান ডিরেক্টরিতে কি কি আছে বিস্তারিত দেখে নেয়া যাক:

```
me@howtocode-pc:~/playground$ ls -l
total 12
drwxrwxr-x 2 me me 4096 Sep  3 12:39 dir1
drwxrwxr-x 2 me me 4096 Sep  3 12:39 dir2
-rw-r--r-- 1 me me 2095 Sep  3 12:44 passwd
```

এবার আসুন, **cp** কমান্ডের **-v** অপশনটি(verbose) ব্যবহার করে দেখি:

```
me@howtocode-pc:~/playground$ cp -v /etc/passwd .
'/etc/passwd' -> './passwd'
```

এবার কিন্তু কোন ফাইল কোথায় কপি করছে তা আমাদের জানিয়েছে। এবার **-i** অপশন(interactive) ব্যবহার করে দেখি:

```
me@howtocode-pc:~/playground$ cp -i /etc/passwd .  
cp: overwrite './passwd'?
```

যেহেতু আমাদের এখানে passwd নামে একটি ফাইল আগে থেকেই ছিল, -i অপশন ব্যবহারের ফলে আমাদের কাছে অনুমতি চাইছে ফাইলটি ওভাররাইড করা হবে কিনা। আমরা 'y' চাপলে ওভাররাইড করবে আর অন্য যেকোনো কিছু চাপলে ওভাররাইড করবে না।

এবার আমরা **mv** কমান্ডের মাধ্যমে passwd ফাইলটির নাম পরিবর্তন করে fun নাম দিই:

```
me@howtocode-pc:~/playground$ mv passwd fun
```

আসুন এবার fun নামের ফাইলটিকে সবগুলো ডিরেক্টরিতে মুভ করে আবার আগের জায়গায় ফেরত আনি। প্রথমে মুভ করি dir1 ডিরেক্টরিতে:

```
me@howtocode-pc:~/playground$ mv fun dir1
```

এবার dir1 ডিরেক্টরি থেকে fun ফাইলটি dir2 ডিরেক্টরিতে মুভ করি:

```
me@howtocode-pc:~/playground$ mv dir1/fun dir2
```

এবার dir2 থেকে বর্তমান ডিরেক্টরি অর্থাৎ ফাইলটির প্রাথমিক অবস্থানে মুভ করি:

```
me@howtocode-pc:~/playground$ mv dir2/fun .
```

এবার fun ফাইলটির একটি করে হার্ডলিঙ্ক তৈরী করা যাক বর্তমান ডিরেক্টরি ও dir1 ও dir2 ডিরেক্টরিতে:

```
me@howtocode-pc:~/playground$ ln fun fun-hard  
me@howtocode-pc:~/playground$ ln fun dir1/fun-hard  
me@howtocode-pc:~/playground$ ln fun dir2/fun-hard
```

এবার **ls -l** দিয়ে আমাদের playground এর বর্তমান অবস্থা জেনে নেয়া যাক:

```
me@howtocode-pc:~/playground$ ls -l  
total 16  
drwxrwxr-x 2 me me 4096 Sep  3 13:15 dir1  
drwxrwxr-x 2 me me 4096 Sep  3 13:15 dir2  
-rw-r--r-- 4 me me 2095 Sep  3 12:44 fun  
-rw-r--r-- 4 me me 2095 Sep  3 12:44 fun-hard
```

লক্ষ্য করুন, ২য় কলামে fun ও fun-hard উভয়ের একই সংখ্যা অর্থাৎ 4 আছে। যার অর্থ হচ্ছে ওই ফাইলটির মোট চারটি হার্ডলিঙ্ক আছে। তাছাড়া ফাইলসাইজের জায়গায় উভয়ের সাইজ একই। তাহলে উভয়ই কি একই ফাইল? নাও হতে পারে। এটা বের করতে আরো গভীরে যেতে হবে।

যখন হার্ডলিঙ্কের কথা ভাবছি তখন কল্পনা করুন যে একটা ফাইলের দুটো অংশ হয়। হার্ডডিস্কে ব্লকের পর ব্লক তথ্য সাজিয়ে তথ্যের অংশ যাকে ইনোড(inode) বলে। এবং একটি নামের অংশ যা ওই ইনোডের সাথে আবদ্ধ। আমরা যখন হার্ডলিঙ্কিং করি তখন আসলে ওই ফাইলের ইনোডের জন্য আরো একটি নতুন নাম দিই। অর্থাৎ দুটি ফাইলের ইনোড একই হলে ফাইল দুটি আসলে একটাই ফাইল।

ইনোড নম্বর দেখতে **ls -li** এর সাথে **-li** অপশনটি যুক্ত করতে হয়। সংক্ষেপে **ls -li**। আসুন ব্যবহার করে দেখি:

```
me@howtocode-pc:~/playground$ ls -li
total 16
811034 drwxrwxr-x 2 me me 4096 Sep  3 13:15 dir1
811035 drwxrwxr-x 2 me me 4096 Sep  3 13:15 dir2
794647 -rw-r--r-- 4 me me 2095 Sep  3 12:44 fun
794647 -rw-r--r-- 4 me me 2095 Sep  3 12:44 fun-hard
```

এখানে প্রথম কলামে ইনোড নম্বর আছে। লক্ষ্য করুন fun এবং fun-hard উভয়ের ইনোড নম্বর একই। অর্থাৎ তারা একই ফাইল।

এবার সিমবোলিক লিঙ্ক তৈরি করা যাক। হার্ডলিঙ্কের মতই বর্তমান ডিরেক্টরিতে একটা এবং dir1 ও dir2তে একটা করে:

```
me@howtocode-pc:~/playground$ ln -s fun fun-sym
me@howtocode-pc:~/playground$ ln -s ../fun dir1/fun-sym
me@howtocode-pc:~/playground$ ln -s ../fun dir2/fun-sym
```

প্রথম কমান্ডটি হার্ডলিঙ্ক তৈরি করার মতই শুধু **-s** অপশনটি যুক্ত করা হয়েছে সিমবোলিক লিঙ্ক তৈরি করতে। কিন্তু দ্বিতীয় ও তৃতীয় কমান্ডে fun এর জায়গায় ../fun কেন? কারন dir1 বা dir2 এর সাপেক্ষে ওটাই fun ফাইলটির অবস্থান। অর্থাৎ, dir1 বা dir2 এর সাপেক্ষে fun এর রিলেটিভ পাথ ../fun

আমরা fun ফাইলটির এ্যাবসলিউট পাথ ব্যবহার করেও সিমবোলিক লিঙ্ক তৈরি করতে পারি:

```
me@howtocode-pc:~/playground$ ln -s /home/me/playground/fun dir1/fun-sym
```

আমরা চাইলে dir1 এর সিমবোলিক লিঙ্ক তৈরি করতে পারি:

```
me@howtocode-pc:~/playground$ ln -s dir1 dir1-sym
```

এবার কিছু ডিলিট করা বা মুছে ফেলা যাক। আমরা fun-hard নামের হার্ডলিঙ্কটি মুছে ফেলি প্রথমে:

```
me@howtocode-pc:~/playground$ rm fun-hard
```

এবার **ls -l** কমান্ড দিয়ে দেখি মুছে গেছে কিনা:

```
me@howtocode-pc:~/playground$ ls -l
total 12
drwxrwxr-x 2 me me 4096 Sep  3 13:55 dir1
lrwxrwxrwx 1 me me    4 Sep  3 14:07 dir1-sym -> dir1
drwxrwxr-x 2 me me 4096 Sep  3 13:56 dir2
-rw-r--r-- 3 me me 2095 Sep  3 12:44 fun
lrwxrwxrwx 1 me me    3 Sep  3 13:54 fun-sym -> fun
```

এখানে fun-hard ফাইলটি নেই, অর্থাৎ সেটি মুছে গেছে।

এবার -i অর্থাৎ ইন্টারএ্যাকটিভ অপশনসহ fun ফাইলটি মুছে ফেলার চেষ্টা করি:

```
me@howtocode-pc:~/playground$ rm -i fun
rm: remove regular file 'fun'?
```

এবার fun ফাইলটি মোছার আগে অনুমতি চাইছে। আমরা y চেপে এন্টার দিলে ফাইলটি মুছে গেলো। এবার **ls -l** দিয়ে দেখি:

```
me@howtocode-pc:~/playground$ ls -l
total 8
drwxrwxr-x 2 me me 4096 Sep  3 13:55 dir1
lrwxrwxrwx 1 me me    4 Sep  3 14:07 dir1-sym -> dir1
drwxrwxr-x 2 me me 4096 Sep  3 13:56 dir2
lrwxrwxrwx 1 me me    3 Sep  3 13:54 fun-sym -> fun
```

fun ফাইলটি নেই। এবং হয়ত লক্ষ্য করেছেন আপনার টার্মিনালে **fun-sym -> fun** লাল অক্ষরে লেখা। যার অর্থ ওই সিমবোলিকলিঙ্কটি ভেঙে গেছে বা ব্রোকেন। মানে লিঙ্কটি আর কাজ করবে না। এবার আমরা সিমবোলিক লিঙ্কগুলো মুছে ফেলি:

```
me@howtocode-pc:~/playground$ rm dir1-sym fun-sym
```

এবার আমরা হোম ডিরেক্টরিতে যাই এবং playground ডিরেক্টরিকেই মুছে ফেলি:

```
me@howtocode-pc:~/playground$ cd ~
me@howtocode-pc:~/playground$ rm -r playground
```

লক্ষ্য করুন, playground ফোল্ডারের মধ্যে তখনও কিছু কন্টেন্ট ছিল তাই মুছে ফেলতে আমাদের **-r** অপশনটি ব্যবহার করতে হয়েছে।



## অধ্যায় - তিন

# রিডিরেকশন(Ridirection)

রিডিরেকশন বা I/O redirection('I/O' এর অর্থ হচ্ছে input/output) সম্ভবত শেলের সবচেয়ে বড় ক্ষমতা। আমরা ইনপুট ও আউটপুট ফাইল থেকে নিতে পারি বা ফাইলে পাঠাতে পারি এমনকি একটা কমান্ডের আউটপুট অন্য একটি কমান্ডের ইনপুট হিসেবেও ব্যবহার করতে পারি পাইপলাইন(pipeline) এর মাধ্যমে।

- স্ট্যান্ডার্ড ইনপুট, আউটপুট এবং এরর: স্ট্যান্ডার্ড ইনপুট, আউটপুট এবং এরর সম্পর্কে প্রাথমিক ধারণা।
- স্ট্যান্ডার্ড আউটপুট রিডিরেকশন: স্ট্যান্ডার্ড আউটপুট রিডিরেকশনের পদ্ধতি।
- স্ট্যান্ডার্ড এরর রিডিরেকশন: স্ট্যান্ডার্ড এরর রিডিরেকশনের পদ্ধতি।
- স্ট্যান্ডার্ড আউটপুট ও এরর একত্রে রিডিরেকশন: স্ট্যান্ডার্ড এরর ও আউটপুট একই ফাইলে রিডিরেকশন ও ডিসপোজের পদ্ধতি।
- ফাইল সংযুক্তিকরণ: **cat** কমান্ড এর ব্যবহার।
- পাইপলাইন: পাইপলাইন, tee ও বিভিন্ন ফিল্টারের ব্যবহার।

## স্ট্যান্ডার্ড ইনপুট, আউটপুট এবং এরর

বিভিন্ন প্রোগ্রাম বিভিন্ন ধরনের আউটপুট আমাদের দেখায়। আমরা যখন `ls` কমান্ড ব্যবহার করি তখন তার আউটপুট দেখি স্ক্রীনে। আবার কখনো কোনো এরর হলে সেটিও দেখতে পাই। ইউনিক্স দুনিয়ায় একটি কথা প্রচলিত যে, "সবকিছুই ফাইল।" আউটপুট এবং এরর দুটোই `stdout` ও `stderr` নামে দুটো বিশেষ ফাইলে পাঠানো হয়। এই ফাইলদুটো স্ক্রীনের সাথে লিঙ্ক করা হয় এবং সেভ করা হয় না।

অন্যদিকে অনেক অধিকাংশ প্রোগ্রাম ইনপুট নেয় ব্যবহারকারীর কাছ থেকে। ইনপুট নেয়ার ক্ষেত্রে কীবোর্ডকে স্ট্যান্ডার্ড ইনপুট হিসেবে ব্যবহার করা হয়।



## স্ট্যান্ডার্ড আউটপুট রিডিবেকশন

আমরা আগের লেসনে জেনেছি স্ট্যান্ডার্ড আউটপুটকে stdout নামে একটি ফাইলে পাঠানো হয়। আমরা টার্মিনালে যা দেখি তা সেখান থেকেই আসে। এখন আমরা যদি সেটা stdout না পাঠিয়ে অন্য কোনো ফাইল বা প্রোগ্রামে পাঠাতে চাই সেটা সম্ভব। প্রথমে আমরা সাধারণ একটা কমান্ডে কি হয় দেখি:

```
me@howtocode-pc:~$ ls -l /usr/bin/
total 282068
-rwxr-xr-x 1 root root      39552 Mar 24 13:35 [
lrwxrwxrwx 1 root root         8 Jul  9 03:51 2to3 -> 2to3-2.7
-rwxr-xr-x 1 root root       96 Mar 23 05:55 2to3-2.7
-rwxr-xr-x 1 root root       96 Apr 11 20:14 2to3-3.4
-rwxr-xr-x 1 root root    10320 Feb  7 2013 411toppm
-rwxr-xr-x 1 root root       39 Feb 18 2012 7z
-rwxr-xr-x 1 root root       40 Feb 18 2012 7za
-rwxr-xr-x 1 root root   106296 Mar 28 00:52 a2p
lrwxrwxrwx 1 root root        25 Aug 21 14:15 aclocal -> /etc/alternatives/aclocal
-rwxr-xr-x 1 root root    36792 Jan  3 2014 aclocal-1.14
-rwxr-xr-x 1 root root    19008 Jan 17 2014 aconnect
-rwxr-xr-x 1 root root    15008 Apr  4 00:41 acpi_listen
-rwxr-xr-x 1 root root     6123 Apr 30 22:59 add-apt-repository
-rwxr-xr-x 1 root root     6280 Jun  4 02:54 addpart
-rwxr-xr-x 1 root root    27696 Apr 16 01:00 addr2line
-rwxr-xr-x 1 root root    73184 Jan 17 2014 alsaloop
-rwxr-xr-x 1 root root    65456 Jan 17 2014 alsamixer
-rwxr-xr-x 1 root root    15072 Jan 17 2014 alsaucm
-rwxr-xr-x 1 root root    19008 Jan 17 2014 amidi
-rwxr-xr-x 1 root root    52664 Jan 17 2014 amixer
.....
```

আপনি দীর্ঘ একটি লিস্ট দেখবেন এর /usr/bin/ এর কন্টেন্ট এর। অনেক সময় লাগবে টার্মিনালে স্ক্রল করে করে দেখতে। জায়গাও নষ্ট হবে প্রচুর। আমরা যা করতে পারি তা হলো এই আউটপুট একটা ফাইলে সংরক্ষণ বা সেভ করে রাখা। পরে সেটা দেখতে পারি **less** কমান্ড দিয়ে অনেক কম ঝামেলাবিহীন উপায়ে। তারমানে, আমরা এই আউটপুট বা স্ট্যান্ডার্ড আউটপুট যা স্ক্রীনে আসছে তাকে একটা ফাইলে রিডিবেক্ট করে দেবো। কিন্তু কিভাবে?

রিডিবেক্ট এর জন্য কমান্ড নেই কোনো বরং একটি শেল ফিচার আছে যার মাধ্যমে '>' চিহ্নের মাধ্যমে আমরা সহজেই আউটপুট রিডিবেক্ট করতে পারি। আমরা যদি উপরের এই কমান্ডটির আউটপুট **ls-output.txt** নামের একটি ফাইলে রিডিবেক্ট করতে চাই তো লিখতে হবে:

```
me@howtocode-pc:~$ ls -l /usr/bin/ > ls-output.txt
```

কমান্ডটি দেয়ার পর আপনি স্ক্রীনে কিছুই দেখবেন না, কোনো দীর্ঘ লিস্টই নেই। বরং আপনার ওয়াকিং ডিরেক্টরিতে ls-output.txt নামে একটি ফাইল পাবেন।

```
me@howtocode-pc:~$ ls -l ls-output.txt
-rw-rw-r-- 1 me me 140433 Sep  5 20:42 ls-output.txt
```

দেখুন, ১৪০ কিলোবাইটের দীর্ঘ টেক্সট ফাইল। আপনি `less ls-output.txt` কমান্ডের মাধ্যমে দেখতে পারবেন ফাইলে কি কি লেখা আছে।

আমরা যদি **ls-output.txt** ফাইলে আরো কিছু আউটপুট একইভাবে রিডিবেইট করতে চাই সেক্ষেত্রে একটা ঘটনা ঘটবে। তা হচ্ছে আগের তথ্য সব মুছে যাবে। কিন্তু আমরা যদি তা না চাই, যদি চাই যে আগের তথ্যও থাকুক এবং তারসাথে নতুন তথ্য যোগ হোক তাহলে `>` এর পরিবর্তে `>>` ব্যবহার করতে হবে।

```
me@howtocode-pc:~$ ls -l ls-output.txt
-rw-rw-r-- 1 me me 280866 Sep  6 15:42 ls-output.txt
```

দেখুন, আমরা আগের কমান্ডটিই দিয়েছি কিন্তু এবার `>>` দিয়ে। ফলে আগের তথ্যের সাথে নতুন রিডিবেইট করা তথ্য মিলে দ্বিগুন তথ্য এসেছে।

## স্ট্যান্ডার্ড এরর রিডিবেকশন

একটা প্রোগ্রাম একাধিক স্ট্রীম বা চ্যানেলে তথ্য আদানপ্রদান করে। এই স্ট্রীমগুলোকে ব্যবহারিক প্রয়োজনে নাম বা সংখ্যা দিয়ে প্রকাশ করা হয় যাকে ফাইল ডেস্ক্রিপ্টর(file descriptor) বলে। শেল প্রোগ্রামের প্রথম তিনটি স্ট্রীমকে ডেস্ক্রিপ্টর 0, 1 ও 2 নাম দেয়। এরা যথাক্রমে ইনপুট, আউটপুট ও এরর এর স্ট্রীম। স্বাভাবিকভাবে '>' চিহ্ন ডেস্ক্রিপ্টর 1 রিডিবেক্ট করে কিন্তু আমরা এর সাথে এরর ডেস্ক্রিপ্টরের সংখ্যা যোগ করে অর্থাৎ '2>' চিহ্ন দিয়ে এরর রিডিবেক্ট করতে পারি।

আমরা যদি /bin/usr বলে কোনো ডিরেক্টরি না তৈরি করে থাকি তবে ডিফল্টভাবে এই ডিরেক্টরি আপনি সিস্টেমে পাবেন না। তাই এটাকে **ls -l** কমান্ডের সাথে ব্যবহার করলে এরর দেখাবে। আমরা এই এররটি ls-error.txt নামের একটি ফাইলে রিডিবেক্ট করছি:

```
me@howtocode-pc:~$ ls -l /bin/usr/ 2> ls-error.txt
me@howtocode-pc:~$ less ls-error.txt
ls: cannot access /bin/usr/: No such file or directory
(END)
```

রিডিবেক্ট করার ফলে স্ক্রীনে আমরা কোনো এররই দেখাবে না বরং আমরা `less ls-error.txt` কমান্ড দিয়ে ফাইলটি খুললে এরর মেসেজটি পাবো।

## স্ট্যান্ডার্ড আউটপুট ও এর একত্রে রিডিরেকশন

আমরা পূর্ববর্তী লেসনদুটোয় দেখেছি কীভাবে স্ট্যান্ডার্ড আউটপুট এবং স্ট্যান্ডার্ড এরর কোনো ফাইলে রিডিরেক্ট করতে হয়। আমরা এবার দেখবো কীভাবে স্ট্যান্ডার্ড আউটপুট ও এরর উভয়কেই একই ফাইলে রিডিরেক্ট করতে হয়। এর দুটো পদ্ধতি আছে। একটি শেলের পুরনো ভার্সনগুলোর জন্য। হয়ত আপনাকে কখনোই এটি ব্যবহার করতে হবে না। তবুও দেখে রাখা যাক। মনে করুন `ls -l /bin/usr` (যদিও `/bin/usr/` বলে কোনো ডিরেক্টরি নেই।) কমান্ডটির আউটপুট ও এরর `ls-output.txt` ফাইলে রিডিরেক্ট করবো। সেক্ষেত্রে আমাদের লিখতে হবে:

```
me@howtocode-pc:~$ ls -l /bin/usr > ls-output.txt 2>&1
```

`ls -l /bin/usr > ls-output.txt` এই অংশটুকু দিয়ে আমরা কমান্ডটির আউটপুট **ls-output.txt** ফাইলে রিডিরেক্ট করেছি এবং তারপর `2>&1` দিয়ে আমরা নির্দেশ দিয়েছি ফাইল ডেস্ক্রিপ্টর ২ কে ফাইল ডেস্ক্রিপ্টর ১ এ রিডিরেক্ট করতে অর্থাৎ স্ট্যান্ডার্ড এররকেই আমরা স্ট্যান্ডার্ড আউটপুটে রিডিরেক্ট করেছি এবং স্ট্যান্ডার্ড আউটপুট যেহেতু `ls-output.txt` ফাইলে রিডিরেক্ট করা তাই এররগুলোও ওখানেই জমা হবে।

আধুনিক শেল আমাদের আর ভালো একটা উপায় দেয় উভয়কে একই ফাইলে রিডিরেক্ট করতে:

```
me@howtocode-pc:~$ ls -l /bin/usr &> ls-output.txt
```

অর্থাৎ, রিডিরেকশনের চিহ্ন হিসেবে **&>** ব্যবহার করলেই হবে। আমরা আগে থেকেই থাকা কোনো ফাইলে আরো এরকম রিডিরেক্ট করতে পারি **&>>** চিহ্ন দিয়ে।

টিপস: কখনো কখনো আপনার মনে হতেই পারে এত আউটপুট বা এরর কিছুই আপনি দেখতে চান না। আপনি তখন সমস্তকিছু **/dev/null** এ রিডিরেক্ট করে দিতে পারেন। তখন সেগুলো না স্ক্রীনে জমা হবে না সেভ হবে কোনো ফাইলে। এটা করতে পারেন এভাবে:

```
me@howtocode-pc:~$ ls -l /bin/usr/ &> /dev/null
```

**/dev/null** একটি বিশেষ ফাইল যা ইনপুট নেয় কিন্তু তা নিয়ে কিছুই করে না। এই ফিচারটিকে বিটবাকেট বলা হয়।

## ফাইল সংযুক্তিকরণ

এবার আমরা স্ট্যান্ডার্ড ইনপুট নিয়ে কাজ করে এমন একটি কমান্ড নিয়ে কাজ করবো। কমান্ডটি হল **cat**। এই কমান্ডটি যেকোনো ফাইল এর কন্টেন্ট স্ট্যান্ডার্ড ইনপুট হিসেবে ব্যবহার করে এবং স্ট্যান্ডার্ড আউটপুটে সেটাকে রিডিরেক্ট করে। এর কমান্ড কাঠামোটি এরকম:

```
cat [file...]
```

আমরা একাধিক ফাইলকে **cat** এর আর্গুমেন্ট হিসেবে ব্যবহার করতে পারি। সেক্ষেত্রে **cat** তাদের পরপর জুড়ে দিয়ে একসাথে আউটপুট দেবে। প্রকৃতপক্ষে এই কমান্ডটির কাজই হল ফাইল জুড়ে দেয়া। আসুন দেখে নেয়া যাক:

```
me@howtocode-pc:~$ ls e-books/ > ~/ls-e-books.txt
me@howtocode-pc:~$ ls Music/ > ls-music.txt
me@howtocode-pc:~$ cat ls-e-books.txt
a_game_as_old_as_empire_hxforum_org_sophie.pdf
BLFS-BOOK-7.5-nochunks.pdf
Craft and Skills
debian-handbook.pdf
Full_Circle
Harry Potter Series
JohnPerkins-ConfessionsOfAnEconomicHitman.pdf
LFS-BOOK-7.5.pdf
Religious
Sherlock Holmes
sherlock.pdf
tarashankar_bandyopadhyay_kobi.pdf
Unsorted
আফরোজা পারভিন
কবীর মুন্সী
গল্পগ্রন্থ-কমলকুমার মজুমদার.pdf
প্রণব ডাউ
বুদ্ধদেব গুহ
বুদ্ধদেব বসু
মাইকেল মধুসূদন দত্ত
মীর মোশাররফ হোসেন
মুহম্মদ জাফর ইকবাল
রবীন্দ্রনাথ ঠাকুর
রবীন্দ্রনাথ ঠাকুর_
বুপ গ্রাহিত্য
মুকুমার রায়
মুনীন গোস্বামী
হুমায়ূন আহমেদ
হুমায়ূন আজাদ
me@howtocode-pc:~$ cat ls-music.txt
Bob Dylan
Music
```

```

Pete Seeger - The Essential Pete Seeger (2005)
Ringtones
Soulful-Voice-Arjit-Singh-128Kbps-2014(Songs.PK)
Sufi
Veer
জল এর গান
জাতিস্মর
বতন দা
ববীন্দ্রসহীত
me@howtocode-pc:~$ cat ls-e-books.txt ls-music.txt
a_game_as_old_as_empire_hxforum_org_sophie.pdf
BLFS-BOOK-7.5-nochunks.pdf
Craft and Skills
debian-handbook.pdf
Full_Circle
Harry Potter Series
JohnPerkins-ConfessionsOfAnEconomicHitman.pdf
LFS-BOOK-7.5.pdf
Religious
Sherlock Holmes
sherlock.pdf
tarashankar_bandyopadhyay_kobi.pdf
Unsorted
আফরোজা পারভিন
কবীর মুন্সী
গল্পসমগ্র-কমলকুমার মজুমদার.pdf
প্রণব ডে
বুদ্ধদেব গুহ
বুদ্ধদেব বসু
মাটেকেল মধুসূদন দত্ত
মীর মোপাররফ হোসেন
মুহম্মদ জাফর ইকবাল
ববীন্দ্রনাথ ঠাকুর
ববীন্দ্রনাথ ঠাকুর_
বুপ্ৰা হিঅ
মুকুমার রায়
মুনীল গঙ্গোপাধ্যায়
হুমায়ূন আহমেদ
হুমায়ূন আজাদ
Bob Dylan
Music
Pete Seeger - The Essential Pete Seeger (2005)
Ringtones
Soulful-Voice-Arjit-Singh-128Kbps-2014(Songs.PK)
Sufi
Veer
জল এর গান
জাতিস্মর
বতন দা
ববীন্দ্রসহীত

```

প্রথমে আমরা `ls e-books/ > ls-e-books.txt` কমান্ড দিয়ে **e-books** ফোল্ডারের কন্টেন্ট এর লিস্ট `ls-e-books.txt` ফাইলে রিডিবেক্ট করে সংরক্ষণ করলাম। একইভাবে `ls Music/ > ls-music.txt` কমান্ড দিয়ে **Music** ফোল্ডারের কন্টেন্টের লিস্ট সংরক্ষণ করলাম `ls-music.txt` ফাইলে।

এর পরেরন কমান্ডদুটি অর্থাৎ `cat ls-e-books.txt` এবং `cat ls-music.txt` কমান্ড দিয়ে ওই ফাইলগুলোর কন্টেন্ট আলাদাভাবে দেখিয়েছি।

সবশেষে `cat ls-e-books.txt ls-music.txt` কমান্ড দিয়েছি। ফলে দুটি ফাইলের কন্টেন্ট জুড়ে দিয়ে স্ক্রীনে দেখিয়েছে। আমরা চাইলে এই আউটপুট '>' দিয়ে এই আউটপুট অন্য কোনো ফাইলে রিডিবেক্ট করে দিতে পারতাম।

মজার বিষয় হচ্ছে **cat** শুধু টেক্সট না যেকোনো কিছু জুড়ে দিতে পারে। মনে করুন আপনি ইন্টারনেট থেকে একটি মুভি নামিয়েছেন। যে আপলোড করেছে সে এটিকে ছোট ছোট ভাগ করে আপলোড করেছে। আপনাকে সবগুলো নামিয়ে করে নিতে হবে। মনে করি এরকম ছোট ছোট ১৫টা ফাইল আছে। `movie.mpeg.001`, `movie.mpeg.002....` এরকম করে `movie.mpeg.015` পর্যন্ত। আমরা এগুলো পরপর জুড়ে দিয়ে `movie.mpeg` ফাইল বানাতে পারি এভাবে:

```
me@howtocode-pc:~$ cat movie.mpeg.0* > movie.mpeg
```

এখানে আমরা সবফাইলগুলো "ওয়াইল্ডকার্ডের মাধ্যমে সিলেক্ট করেছি আর ওয়াইল্ডকার্ড সবসময়ই ক্রম বজায় রাখে অর্থাৎ এটা `001`, `002....` `015` এভাবেই সিলেক্ট করবে। \*`cat` দিয়ে এদের জুড়ে দিয়েছি এবং এর আউটপুট '>' চিহ্ন দিয়ে `movie.mpeg` ফাইলে রিডিবেক্ট করে দিয়েছি।

আমরা যদি কোনো আর্গুমেন্ট ছাড়া **cat** কমান্ডটি ব্যবহার করি এভাবে:

```
me@howtocode-pc:~$ cat
```

তাহলে আপনি দেখবেন প্রম্পট ফিরে আসছে না। দেখে মনে হতে পারে হ্যাং হয়ে গেছে। আসলে তা হয়নি। **cat** তখন আসলে আপনার কাছ থেকে ইনপুট আশা করছে। আপনি যেকোনোকিছু লিখে এন্টার চাপলে সেটা `cat` স্ট্যান্ডার্ড আউটপুটে পাঠাবে। এরকম:

```
me@howtocode-pc:~$ cat
আমি সব দেখে গুলুন ফেপে গিয়ে করি বাস্তবায় চিংকার!
আমি সব দেখে গুলুন ফেপে গিয়ে করি বাস্তবায় চিংকার!
```

সবশেষে আপনি `ctrl-d` চেপে এখান থেকে বেরতে পারবেন।

আমরা চাইলে এই আউটপুট কোনো ফাইলে রিডিবেক্ট করতে পারি। তখন নীচে আউটপুট না দেখিয়ে সেই ফাইলে পাঠিয়ে দেব:

```
me@howtocode-pc:~$ cat > চিংকার.txt
আমি সব দেখে গুলুন ফেপে গিয়ে করি বাস্তবায় চিংকার!
```

আবার আমরা সেই ফাইলটার কন্টেন্ট ইনপুট হিসেবে `cat` এ দিতে পারি '<' চিহ্ন দিয়ে:

```
me@howtocode-pc:~$ cat < চিংকার.txt
```

আমি সব দৈবে শূনে কৈলে গিয়ে কৰি বাঙলায় চিংকার!



# পাইপলাইন

আমরা এপর্যন্ত ফাইল থেকে কমান্ড ও কমান্ড থেকে ফাইলে আউটপুট, এরর বা ইনপুট রিডিরেক্ট করেছি। এবার আমরা দেখবো একটি কমান্ডের স্ট্যান্ডার্ড আউটপুট কিভাবে অন্য একটি কমান্ড ইনপুট হিসেবে সরাসরি ব্যবহার করতে পারে। এই কাজটি করতে শেলে যে ফিচারটা থাকে তাকে বলে পাইপলাইন।

পাইপলাইন ব্যবহার করতে হয় '|' চিহ্ন দিয়ে। এর ব্যবহারিক কাঠামোটি এমন:

```
command1 | command2
```

আমরা এর আগে(1.3.2.stdordrct.md) `ls -l /usr/bin` কমান্ডের আউটপুট `ls-output.txt` এ রিডিরেক্ট করে `less` কমান্ড দিয়ে সেটা দেখেছি। পাইপলাইন ব্যবহার করে আমরা সহজেই কোনো ফাইলে এই আউটপুট রিডিরেক্ট না করে সরাসরি `less` কমান্ডেই রিডিরেক্ট করে দেখতে পারি এভাবে:

```
me@howtocode-pc:~$ ls -l /usr/bin/ | less
```

কমান্ডটি ব্যবহার করলে কোনো ফাইলে রিডিরেকশন ছাড়াই সরাসরি আপনাকে `less` এর মাধ্যমে স্ক্রীনে `ls -l /usr/bin` কমান্ডের আউটপুট দেখাবে।

## ফিল্টার

বেশ কিছু কমান্ড আছে যেগুলো ব্যবহার করে আমরা আউটপুটকে সাজাতে পারি কোনো বিশেষ নিয়মে। বা শুধু বিশেষ কোনো তথ্য বের করে আনতে পারি। এদের ফিল্টার বলে। আমরা এখন কয়েকটা ফিল্টার নিয়ে আলোচনা করবো।

## sort

`sort` কমান্ডটি দিয়ে আমরা কোনো কমান্ডের আউটপুট বা টেক্সটফাইলের কমান্ডগুলোকে বিশেষ ক্রমে সাজাতে পারি। ডিফল্টভাবে এটি বর্ণানুক্রমিকভাবে সাজায়। একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ ls /usr/bin/ /bin/ | sort | less
```

স্বাভাবিকভাবে যদি আমরা `ls /usr/bin /bin/` কমান্ডটি দিতাম তবে `/usr/bin/` ও `/bin/` ডিরেক্টরির কন্টেন্টের দুটি পৃথক লিস্ট আমরা পেতাম। আমরা যে আউটপুট পেতাম পাইপলাইন দিয়ে সেটা `sort` কমান্ডের কাছে পাঠানোর ফলে `sort` কমান্ড লিস্টদুটির আইটেমগুলি নিয়ে বর্ণানুক্রমে সাজিয়ে একটি লিস্ট বানিয়েছে। এই নতুন অখণ্ড লিস্টকে আমরা আবার পাইপলাইনের মাধ্যমে `less` কমান্ড দিয়ে দেখেছি।

## uniq

**uniq** কমান্ড দিয়ে ইউনিক আইটেম খুঁজে নিতে হয়। যেমন যদি আমরা আগের কমান্ডটি দিই বেশ কিছু ফাইল পাওয়া যাবে যা `/usr/bin/` ও `/usr/` উভয় ডিরেক্টরিতে আছে। আমরা যদি **sort** এর সাথে **uniq** কমান্ড পাইপলাইনে যোগ করি তবে যেসব আইটেম দুবার আছে তার থেকে একটা দেখাবে। এটা আমরা করতে পারি এভাবে:

```
me@howtocode-pc:~$ ls /usr/bin/ /bin/ | sort | uniq | less
```

আবার আমরা যদি চাই শুধু সেই সব আইটেমই দেখাবো যা উভয় ডিরেক্টরিতেই আছে তবে তারজন্য আমরা **uniq -d** ব্যবহার করতে পারি এভাবে:

```
me@howtocode-pc:~$ ls /usr/bin/ /bin/ | sort | uniq -d | less
```

## WC

**wc** কমান্ডটি দিয়ে কোনো আউটপুট বা টেক্সটের শব্দসংখ্যা গোনা যায়। আমরা যদি আগের কমান্ডটার শেষে **less** না দিয়ে **wc -l** কমান্ড ব্যবহার করি তাহলে আমরা শব্দসংখ্যা জানতে পারবো। স্ট্যান্ডার্ড ইনপুট থেকে গুনতে **-l** অপশন ব্যবহার করতে হয়। যেমন:

```
me@howtocode-pc:~$ ls /usr/bin/ /bin/ | sort | uniq | wc -l
2392
```

## grep

**grep** কমান্ডটি ব্যবহার করা হয় টেক্সটের মধ্য থেকে বিশেষ প্যাটার্ন খুঁজে বের করে। এর কমান্ড কাঠামোটি এরকম:

```
grep pattern [file...]
```

এই কমান্ডটি পাইপলাইনে জুড়ে দিয়ে আমরা প্রয়োজনীয় জিনিস খুঁজে নিতে পারি। `/usr/bin` ও `/bin` এর যে অখন্ড সার্ভেড ইউনিক লিস্ট আমরা বানিয়েছিলাম তার মধ্য থেকে যেগুলোর নামের ভেতর 'zip' কথাটি আছে সেগুলো খুঁজে বের করতে পারি এভাবে:

```
me@howtocode-pc:~$ ls /usr/bin/ /bin/ | sort | uniq | grep zip
bunzip2
bzip2
bzip2recover
funzip
gpg-zip
gunzip
gzip
mzip
preunzip
prezip
prezip-bin
unzip
unzipsfx
zip
zipcloak
zipdetails
zipgrep
zipinfo
zipnote
zipsplit
```

## head এবং tail

**head** এবং **tail** কমান্ড দিয়ে কোনো আউটপুটের যথাক্রমে প্রথম ও শেষ ১০ লাইন দেখা যায়। **-n** অপশন দিয়ে ঠিক করে দেয়া যায় কতগুলো লাইন দেখাবে। `ls /usr/bin/` কমান্ডের আউটপুটের প্রথম ১০ লাইন দেখতে পারি এভাবে:

```
me@howtocode-pc:~$ ls /usr/bin/ | head
[
2to3
2to3-2.7
2to3-3.4
411toppm
7z
7za
a2p
aclocal
aclocal-1.14
```

আবার মনে করুন একই কমান্ডের শেষের লাইন দেখতে চাই। কিন্তু ডিফল্টভাবে ১০টা দেখায় আমরা ৫টা দেখতে চাই। তাহলে লিখবো:

```
me@howtocode-pc:~$ ls /usr/bin/ | tail -n 5
zjsdecode
zlib-flate
zsoelim
zsync
zsyncmake
```

## tee

আমার যারা শহরে থাকি এবং যাদের অটোমেটেড মোটরচালিত পানি সরবরাহের ব্যবস্থা আছে তাদের বাড়িতে পানির পাইপের নেটওয়ার্ক দেখা যায়। আমরা এতক্ষণ যে পাইপলাইনের কথা বলেছি তার সাথে এই পানি সরবরাহের নেটওয়ার্কের তুলনা করা যায়। তবে এতক্ষণ শুধু পাইপের পর পাইপজুড়ে লম্বা পাইপ তৈরি করা গেছে। আমরা পানি সরবরাহের নেটওয়ার্কে প্রায়ই 'T' এর মত দেখতে একধরনে সংযোগস্থল দেখি যেখান থেকে দুদিকে পানি সরবরাহ করা যায়। **tee** কমান্ডটিও একই কাজ করে। এর তিনটি পথ। একটি থেকে ইনপুট নেয় আর অন্য দুটির একটি থেকে আউটপুট টেক্সট ফাইলে পাঠায়। অপরটি দিয়ে স্ক্রীনে তথ্য পাঠায়। স্ক্রীনে পাঠানো তথ্যকে আমরা আবার পাইপলাইনের মাধ্যমে বিভিন্ন ফিল্টারে যুক্ত করতে পারি। একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ ls /usr/bin/ | tee ls.txt | grep zip
funzip
gpg-zip
mzip
preunzip
prezip
prezip-bin
unzip
unzipsfx
zip
zipcloak
zipdetails
zipgrep
zipinfo
zipnote
zipsplit
```

কমান্ডটির শুরুতে আমরা `ls /usr/bin/` কমান্ড দিয়ে `/usr/bin/` ডিরেক্টরি এর কন্টেন্ট এর লিস্ট করেছি। তারপর সেটাকে পাইপলাইনের মাধ্যমে **tee** কমান্ডের সাথে যুক্ত করেছি। **tee** কমান্ডের আর্গুমেন্ট হিসেবে আমরা টেক্সট ফাইল `ls.txt` দিয়েছি। যার ফলে `ls /usr/bin/` এর আউটপুট `ls.txt` ফাইলে সেভ রাখবে। এরপর আমরা **tee** এর সাথে পাইপলাইনে **grep** কমান্ডটির সাথে আর্গুমেন্ট হিসেবে দিয়েছি `zip`। ফলে স্ক্রীনে শুধু সেইসব নাম দেখাবে যার নামের মধ্যে `zip` কথাটি আছে।

## অধ্যায় - চার

# শেলের চোখে দেখা

প্রত্যেক কমান্ডের নিজস্ব ক্ষমতা আছে। আবার শেল স্বয়ং কিছু ক্ষমতার অধিকারি। শেল বিল্টইন কমান্ড, ওয়াইল্ডকার্ড পাইপলাইন সবই শেলের ফিচার। আমরা এগুলো আগের অধ্যায়গুলোতে ব্যবহার করেছি। এই অধ্যায়ে আমরা শেলের চোখে দেখবো। দেখবো কীভাবে শেল এগুলো দেখে, বিচার করে, কার্যকর করে। আরো দেখবো শেলকে কীভাবে আরো স্মুথলি ব্যবহার করা যায়।

- **এক্সপ্যানসন:** এক্সপ্যানসন সম্পর্কিত প্রাথমিক ধারণা।
- **পাথনেম এক্সপ্যানসন:** পাথনেম এক্সপ্যানসন এর ব্যাখ্যা ও ব্যবহার।
- **গাণিতিক এক্সপ্যানসন:** গাণিতিক এক্সপ্যানসন এর ব্যবহার।
- **ব্রেস এক্সপ্যানসন:** ব্রেস এক্সপ্যানসনের ব্যবহার।
- **প্যারামিটার এক্সপ্যানসন:** প্যারামিটার এক্সপ্যানসন সম্পর্কিত প্রাথমিক ধারণা।
- **কমান্ড সাবস্টিটিউশন:** কমান্ড সাবস্টিটিউশনের ব্যবহার।
- **ক্যাটিং:** ক্যাটিং সম্পর্কিত প্রাথমিক ধারণা, ডবল ক্যাট ও সিঙ্গেল ক্যাট এর ব্যবহার।
- **ক্যারেটার স্কেইপিং:** স্কেইপ ক্যারেটারের ব্যবহার।

## শেল এক্সপ্যানসন

আমরা কমান্ড লিখি এবং এন্টার চাপি। শেল সেগুলোকে কার্যকর করে। কিন্তু মজার ব্যাপারটা হল, শেল হুবহু আমাদের কমান্ড কার্যকর করে না বরং অন্তর্বর্তীকালীন কিছু ধাপ সে নিজের মধ্যে অতিদ্রুত করে ফেলে। এই যে ওয়াইল্ডকার্ডের ব্যাপারটাই ধরুন না। শেলের কাছে "চিহ্নটি নিছক একটি চিহ্ন না। আরো অনেক মানে আছে তার। এই সব মানে তৈরী হওয়া বা বোঝার জন্য শেলকে ভেঙে ভেঙে বুঝতে হয়। অর্থাৎ দুর্বল ছাত্রকে যেভাবে সরলীকৃত করে বোঝানো হয়। এই সরলীকরণ এর প্রক্রিয়াকে এক্সপ্যানসন বলা হয়। আমরা একটা উদাহরণ দেখলেই সব পরিষ্কার হয়ে যাবে। উদাহরণটায় আমরা `*echo` নামের একটি কমান্ড ব্যবহার করবো। এই শেল-বিল্টইন কমান্ডটি খুব সাধারণ একটা কাজ করে। এর আর্গুমেন্ট হিসেবে আপনি যা লিখবেন এটি তাই স্ক্রীনে দেখাবে। আসুন, দেখা যাক:

```
me@howtocode-pc:~$ echo বাঙলা আমার তৃষ্ণার জন, তৃপ্ত শেষ চুমুক...  
বাঙলা আমার তৃষ্ণার জন, তৃপ্ত শেষ চুমুক...
```

দেখা গেলো, **echo** এর আর্গুমেন্ট হিসেবে আমরা যা লিখেছি সেটারই পুনরাবৃত্তি করলো কমান্ডটি। আসুন **echo** কে আবার ব্যবহার করে দেখি:

```
me@howtocode-pc:~$ echo *  
8_1.pdf 8_1.pdf.aria2 AioServer3.4.2_portable AioServer3.4.2_portable.zip archives Audiob  
[...]
```

এবার কিন্তু **echo** "প্রিন্ট" করেনি। বরং "কে এক্সপ্যান্ড করে আমার কারেন্ট ওয়ার্কিং ডিরেক্টরি অর্থাৎ হোমের সকল ফাইল ও ফোল্ডারের নাম দেখিয়েছে।

## পাথনেম এক্সপ্যানসন

আমরা পূর্ববর্তী চ্যাপ্টারগুলোতে যে ওয়াইল্ডকার্ড এর ব্যবহার দেখেছি তার পোশাকি নাম হচ্ছে পাথনেম এক্সপ্যানসন(pathname expansion)। মনে করি, এই আমাদের হোম ডিরেক্টরির অবস্থা:

```
me@howtocode-pc:~$ ls
8_1.pdf          land_of_lisp      pvim
8_1.pdf.aria2    ls                python
AioServer3.4.2_portable  ls-e-books.txt   RemoteControlServer
AioServer3.4.2_portable.zip  ls-error.txt     reus
archives         ls-home.txt       sd2bk
Audiobooks       ls-music.txt      sent
bin              ls-output.txt     sh_howtocode
Desktop          ls.txt            spiral
diary            mlterm-3.3.8      Templates
diary~          mlterm-3.3.8.tar.gz  test
Documents        Music             test.html
Downloads        Pictures          test.html~
emacs            playground        test.md
emacs.pdf        playlists         TLCL-13.07.pdf
get?ab=128       Podcasts          txt
hfjava           Porteus-KDE-v3.0.1-i486.iso  ubuntu-gnome-14.04-desktop-amd6
hfjavafinalsamples  precise-5.7.1.iso  Videos
hfjavafinalsamples.zip  precise-5.7.1.iso.aria2  আমাদের কথা
hfpython         Public
```

**echo** কমান্ডের আর্গুমেন্ট হিসেবে কোনো টেক্সট দিলে echo সেটাকে হুবহু আউটপুট দেয়। কিন্তু আমরা যদি echo এর আর্গুমেন্টের মধ্যে ওয়াইল্ডকার্ড ব্যবহার করি তাহলে echo কমান্ডটি **ls** এর মত কাজ করবে:

```
me@howtocode-pc:~$ echo D*
Desktop Documents Downloads
```

**echo** আসলে কি করলো? আসুন একটু ভেবে দেখি।

- মনে রাখতে হবে কমান্ডের কাছে আর্গুমেন্ট কিন্তু শেল পৌঁছে দেয়। তাই শেল প্রথমে আমাদের কমান্ড স্টেটমেন্টটি পরীক্ষা করেছে। শেল দেখেছে যে আমরা আর্গুমেন্ট হিসেবে 'D' ব্যবহার করেছি। এবং " চিহ্নকে ওয়াইল্ডকার্ড হিসেবে চিহ্নিত করেছে। শেল ভেবে নিয়েছে **echo** এর কাছে এই আর্গুমেন্ট এক্সপ্যান্ড করে পাঠানো উচিত।
- এক্সপ্যানশনের সিদ্ধান্ত নেয়ার পর শেল খুঁজে বের করেছে যে আমাদের এখনকার ডিরেক্টরি বা হোমে এই এক্সপ্রেশন অর্থাৎ **D\*** এর সাথে কোনগুলো মেলে। অর্থাৎ কোন আইটেমগুলোর শুরুতে D আছে। শেল তখন Desktop, Documents ও Downloads এই তিনটি আইটেম খুঁজে পেয়েছে।
- শেল ওয়াইল্ডকার্ড এক্সপ্রেশন অনুযায়ী পাওয়া আইটেমগুলো বর্ণানুক্রমে সাজিয়ে **echo** এর জন্য আর্গুমেন্ট তৈরি করলো। আরগুমেন্টটা হল 'Desktop Documents Downloads'। এইপর্যন্ত যা যা হলো তা হলো

এক্সপ্যানসন।

- এবার শেল echo কমান্ডের সাথে আর্গুমেন্ট জুড়ে দিয়ে সত্যিকারের কমান্ডটি তৈরি করলো এরকম: **echo Desktop Documents Downloads**। এবং এটি এক্সিকিউট করলো।
- এতক্ষণ echo কমান্ডটি কিছুই করেনি। বরং যা যা করার শেল নিজেই করেছে। এবার echo তাকে আর্গুমেন্টে দেয়া টেক্সট অর্থাৎ 'Desktop Documents Downloads' স্ক্রীনে প্রিন্ট করে দিলো।

## টিল্ডে এক্সপ্যানসন

পাথনেম এক্সপ্যানসনের আরেকটি বিশেষ রূপ টিল্ডে এক্সপ্যানসন(Tilde expansion)। আপনাদের হয়ত মনে আছে আমরা হোম ডিরেক্টরির বদলে '~' চিহ্ন ব্যবহার করেছি। এটা টিল্ডে এক্সপ্যানসনের উদাহরণ। শুধু '~' ব্যবহার করলে:

```
me@howtocode-pc:~$ echo ~
/home/me
me@howtocode-pc:~$ cd ~/Music/
me@howtocode-pc:~/Music$ pwd
/home/me/Music
```

প্রথমে echo ~ কমান্ড দিয়ে দেখে নিলাম '~' এর মান। শেল '~' কে এক্সপ্যান্ড করে /home/me বানিয়েছে। তারপর আমরা cd ~/Music/ কমান্ড দিয়ে ~/Music ডিরেক্টরিতে চুকে pwd দিয়ে ওয়ার্কিং ডিরেক্টরি দেখলাম /home/me/Music। অর্থাৎ সবক্ষেত্রেই ~ এর মান /home/me। আমরা বলতে পারি '~' চিহ্ন ব্যবহার করলে তা বর্তমান ইউজারের হোম ডিরেক্টরি বুঝায়।

এবার মনে করি nishadsingha বলে একজন ইউজার আছে এই কম্পিউটারে। আমরা তার ফাইলপত্র নাড়াচাড়া করতেও '~' ব্যবহার করতে পারি তার ইউজারনেমের সাথে যুক্ত করে:

```
me@howtocode-pc:~$ echo ~nishadsingha
/home/nishadsingha
me@howtocode-pc:~$ cd ~nishadsingha/Music/
me@howtocode-pc:~/Music$ pwd
/home/nishadsingha/Music
```

প্রথমে আমরা echo ~nishadsingha কমান্ড দিয়েছি। যার ফলে ওই ইউজারের হোম ডিরেক্টরি দেখিয়েছে। এরপর cd ~nishadsingha/Music/ কমান্ড দিয়ে সরাসরি তার Music ফোল্ডারে গিয়েছি। pwd কমান্ডের আউটপুটে তার প্রমাণ মেলে।



## গাণিতিক এক্সপ্যানসন

এপর্যায়ে আমরা শেলের গাণিতিক এক্সপ্যানসন(Arithmetic expansion) এর ব্যাপারে জানবো। ওয়াইন্ডোজের ব্যবহার করলে শেল যেমন আগে তাকে পাথনেম হিসেবে এক্সপ্যান্ড করে তেমনি বিশেষ উপায়ে যদি গাণিতিক এক্সপ্রেসন লেখা হয় তাহলে শেল সেটার উত্তর হিসেব করে দিতে পারে। বিশেষ উপায় বলতে এমনভাবে লেখা যেন শেল বুঝতে পারে আমরা চাইছি সে হিসেব করুক। এই এক্সপ্রেসন এর কাঠামো হবে `$( (expression) )`। অর্থাৎ এক্সপ্রেসনের জায়গায় আমাদের দেয়া অংকটি আর তাকে ঘিরে দিতে হবে দুই দফা ব্রাকেট দিয়ে আর সামনে থাকবে একটি ডলার(\$) চিহ্ন। একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ echo $((2 + 2))
4
```

আমাদের মূল অংক ছিল দুই আর দুই এর যোগফল বের করা। সুতরাং এক্সপ্রেসনটি হল: `2 + 2`। আমরা শেলকে অংকটি করার নির্দেশ দিয়েছি এটাকে বিশেষ কাঠামোতে লিখে। অর্থাৎ `$( (2 + 2) )` লিখে। ফলে শেল অংকটি করে উত্তর বের করেছে `4` এবং এটাকেই `echo` কমান্ডের আর্গুমেন্ট হিসেবে পাঠিয়েছে। ফলে `echo` উত্তরটা প্রিন্ট করেছে।

একটা বিষয় মাথায় রাখতে হবে। শেলের গাণিতিক এক্সপ্যানসন খুবই সীমাবদ্ধ। এটা সবসময় পূর্ণসংখ্যা নিয়ে কাজ করবে। কখনোই দশমিক ভগ্নাংশ নিয়ে এটি কাজ করতে পারে না।

এবার দেখা যাক কোন গাণিতিক চিহ্নগুলো আমরা ব্যবহার করতে পারবো:

চিহ্ন	অর্থ
+	যোগ
-	বিয়োগ
*	গুন
/	ভাগ(লক্ষ্যণীয়, ভাগফল শুধু পূর্ণসংখ্যায় দেখানো হবে)
%	ভাগশেষ (ইংরেজিতে Remainder, পোশাকি নাম মড্যুলো(Modulo))
**	ঘাত(চলতিভাষায় আমরা পাওয়ার বলি। এক্সপোনেনশিয়েশন(Exponentiation) ও বলতে পারেন।)

একাধিক সরল এক্সপ্রেসন দিয়ে আপনি যৌগিক বা নেষ্টেড এক্সপ্রেসন তৈরি করতে পারেন। ৫ এর দ্বিতীয় ঘাত( 5 square) কে ৩ দিয়ে গুন করতে হলে আমরা লিখবো:

```
me@howtocode-pc:~$ echo $(( (5**2) * 3 ))
75
```

এবার ভাগের ব্যাপারে দেখি। ভাগসংক্রান্ত গাণিতিক চিহ্ন বা অপারেটরদ্বয় হলো '/' ও '%'। আমরা ৫কে ২দিয়ে ভাগ করবো দুটো দিয়েই:

```
me@howtocode-pc:~$ echo $((5/2))  
2  
me@howtocode-pc:~$ echo $((5%2))  
1
```

প্রথমে আমরা '/' দিয়ে ভাগ করেছি। আসলে উত্তর হওয়া উচিত 2.5 কিন্তু পূর্ণসংখ্যাই শুধু বিবেচ্য বলে 2 দেখিয়েছে। আর পরেরটা দেখিয়েছে ভাগশেষ।

## ব্রেস এক্সপ্যানসন

সকল এক্সপ্যানসন এর মধ্যে ব্রেস এক্সপ্যানসন(Brace expansion) সম্ভবত সবচেয়ে শক্তিশালী। ব্রেসকে চলতিভাষায় আমরা ব্রাকেট বলি বাঙলায়। এবং তিনধরনের ব্রাকেট বা ব্রেসের মধ্যে সেকেন্ড ব্রাকেট বা কার্লি ব্রেস('{') ব্যবহৃত হবে এখানে। ব্রেস এক্সপ্যানসন ব্যবহার করে আমরা কোনো প্যাটার্নে অনেকগুলো স্ট্রিং তৈরি করতে পারি। একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ echo front-{A,B,C}-back
front-A-back front-B-back front-C-back
```

এক্সপ্যানসনের সময় শুধু ব্রেসের মধ্যবর্তী অংশটুকু এক্সপ্যান্ডেড হবে। তাছাড়া এর সামনে বা পিছনে এমন অংশ জুড়ে দেয়া যায় যা প্রতিটিক্ষেত্রেই একই থাকবে। সামনে এমন কমন অংশ থাকলে এখানে যেমন 'front-' তাকে বলা হয় প্রিমবল(preamble)। একইভাবে পিছনের অংশকে বলে পোস্টস্ক্রিপ্ট বলে(postscript)। আমরা এক্সপ্যানসনের জন্য ব্রেসের মধ্যে অনেককিছুই ব্যবহার করতে পারি। কয়েটা টেক্সট স্ট্রিং কমা দিয়ে আলাদা করে দিতে পারি। তবে মাথায় রাখতে হবে। ব্রেসের মধ্যে কোথাও স্পেস ব্যবহার করা যাবে না। একটা উদাহরণ দেখা যাক স্ট্রিং এর:

```
me@howtocode-pc:~$ echo বাঙলা{দেশ,ভাষা,সাহিত্য}
বাঙলাদেশ বাঙলাভাষা বাঙলাসাহিত্য
```

আমরা প্রিমবল হিসেবে দিয়েছি 'বাঙলা' শব্দটি। আর এক্সপ্যানসনের জন্য ব্রেসের মধ্যে শুধু কমা দিয়ে আলাদা করে তিনটি স্ট্রিং: দেশ, ভাষা ও সাহিত্য। শেল এটাকে এক্সপ্যান্ড করে বাঙলাদেশ, বাঙলাভাষা ও বাঙলাসাহিত্য বানিয়েছে।

তাছাড়াও আমরা নম্বর বা অক্ষরের ক্ষেত্রে রেঞ্জ বলে দিতে পারি। আমরা যদি Number\_1 Number\_2 এভাবে Number\_7 পর্যন্ত দেখতে চাই তাহলে লিখবো:

```
me@howtocode-pc:~$ echo Number_{1..7}
Number_1 Number_2 Number_3 Number_4 Number_5 Number_6 Number_7
```

আমরা Number\_1 থেকে Number\_7 পর্যন্ত চেয়েছি। তাই শুরু হবে 1 দিয়ে ও শেষ হবে 7 দিয়ে। তার মাঝখানে '..'। এটা দিয়ে বোঝানো হলো শুরু(এখানে 1) থেকে শেষ(এখানে 7) পর্যন্ত। আমরা চাইলে 01, 02... এভাবে বা 001, 002 এভাবেও রেঞ্জ দিতে পারি:

```
me@howtocode-pc:~$ echo {01..15}
01 02 03 04 05 06 07 08 09 10 11 12 13 14 15
me@howtocode-pc:~$ echo {001..15}
001 002 003 004 005 006 007 008 009 010 011 012 013 014 015
```

এভাবেই A..Z রেঞ্জ সিলেক্ট করলে অক্ষরগুলো বর্ণানুক্রমে ব্যবহার করবে। আমরা উল্টোদিক থেকেও শুরু করে শুরুতেও আসতে পারি:

```
me@howtocode-pc:~$ echo {Z..A}
Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
```

গাণিতিক এক্সপ্যানসনের মত ব্রেস এক্সপ্যানসন গুলো মিলিয়ে যৌগিক বা নেষ্টেড ব্রেস এক্সপ্যানশনে রূপ দেয়া যায়। মনে করুন আপনার অনেক ছবি আছে সংগ্রহে। একদিন ভাবলেন এলোমেলো করে না রেখে সাল ও মাস অনুযায়ী ফোল্ডার করে রাখবেন। আপনার কাছে ২০১২-২০১৪ সালের ছবি আছে আর আপনি চান photos নামে একটি ফোল্ডার করবেন তারপর তার মধ্যে প্রত্যেক সালের প্রত্যেক মাসের জন্য একটি করে ফোল্ডার করবেন। আপনি অবশ্যই একটি একটি করে ফোল্ডার তৈরি করতে পারেন। কিন্তু তা হবে সময় নষ্ট। আমরা এই কাজটি এভাবে করতে পারি:

```
me@howtocode-pc:~$ mkdir photos
me@howtocode-pc:~$ cd photos/
me@howtocode-pc:~/photos$ mkdir {2012..2014}-{01..12}
me@howtocode-pc:~/photos$ ls
2012-01  2012-05  2012-09  2013-01  2013-05  2013-09  2014-01  2014-05  2014-09
2012-02  2012-06  2012-10  2013-02  2013-06  2013-10  2014-02  2014-06  2014-10
2012-03  2012-07  2012-11  2013-03  2013-07  2013-11  2014-03  2014-07  2014-11
2012-04  2012-08  2012-12  2013-04  2013-08  2013-12  2014-04  2014-08  2014-12
```

আমরা প্রথমে photos নামে একটি ফোল্ডার তৈরি করে তাতে ঢুকেছি। তারপর নেষ্টেড ব্রেস এক্সপ্যানসন ব্যবহার করে সহজেই ফোল্ডার তৈরি করেছি।

## প্যারামিটার এক্সপ্যানসন

প্যারামিটার এক্সপ্যানসন(parameter expansion) নিয়ে আমরা শেল স্ক্রিপ্টিং এর সময় বিস্তারিত আলোচনায় যাবো এখন প্রাথমিক ধারণাটা রাখা যাক।

প্রথমে জানা দরকার ডেরিয়েবল কি। যারা প্রোগ্রামার তারা অবশ্য আগে থেকেই জানেন। সহজভাবে আপনি ডেরিয়েবলকে একটা বাক্স বলতে পারেন যেখানে তথ্য রাখতে পারেন, পরিবর্তন করতে পারেন। আর হাজার হাজার বাক্সের ভীড়ে আপনার বাক্সটি খুঁজে পেতে আপনি একটা নাম দিয়েছেন বাক্সের। এটাই ডেরিয়েবল। একটা নাম এবং তার সাথে পরিবর্তনযোগ্য কিছু তথ্য। পরিবর্তনযোগ্য না হলে কিন্তু সেটা আর ডেরিয়েবল থাকবে না। কন্সট্যান্ট হয়ে যাবে।

সিস্টেম চালু থাকাকালীন বেশকিছু তথ্য এমনি করে বিভিন্ন ডেরিয়েবলে সিস্টেম সংরক্ষণ করে। এই সব ডেরিয়েবলগুলোকে সিস্টেম প্যারামিটার বলে।

আপনি যখন শেলে কোনো শব্দের আগে '\$' চিহ্ন দেবেন শেল সেটাকে ডেরিয়েবল হিসেবে বিবেচনা করবে। যেমন, USER নামের ডেরিয়েবলটি যেটি কিনা বর্তমান ব্যবহারকারীর ইউজারনেম সংরক্ষণ করে, সেটি এক্সপ্যান্ড করে echo কমান্ডে পাঠাতে গেলে আপনাকে লিখতে হবে:

```
me@howtocode-pc:~$ echo $USER  
me
```

আপনি চাইলে সকল এনভায়রনমেন্ট ডেরিয়েবলের লিস্ট দেখতে পারেন **printenv** কমান্ড দিয়ে।

## কমান্ড সাবস্টিটিউশন

কমান্ড সাবস্টিটিউশন(Command Substitution) এর মাধ্যমে আমরা আমরা একটি কমান্ডের আউটপুট এক্সপ্যানসন হিসেবে ব্যবহার করতে পারি। একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ ls -l $(which cp)
-rwxr-xr-x 1 root root 130304 Mar 24 13:35 /bin/cp
```

আসুন দেখা যাক কি হলো। কোনো কমান্ডকে এক্সপ্যানসনের জন্য পাঠাতে গেলে তাকে ব্র্যাকেট দিয়ে আবদ্ধ করে তার সামনে '\$' চিহ্ন দিতে হয়। এখানে যেমন `$(which cp)`। শেল প্রথমে এই কমান্ডটি এক্সিকিউট করেছে আর আউটপুট পেয়েছে `/bin/cp`। এই আউটপুটকে `ls -l` এর আর্গুমেন্ট হিসেবে ব্যবহার করা হয়েছে।

কমান্ড সাবস্টিটিউশনের জন্য অন্যভাবেও লেখা যায়। আমরা উপরের কমান্ডটি এভাবেও লিখতে পারতাম:

```
me@howtocode-pc:~$ ls -l `which cp`
-rwxr-xr-x 1 root root 130304 Mar 24 13:35 /bin/cp
```

## ক্যাটিং

দুটো উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ echo শেষ শব্দটি নেখার আগে আমি অনেকগুলো স্পেস দেবো।
শেষ শব্দটি নেখার আগে আমি অনেকগুলো স্পেস দেবো।
me@howtocode-pc:~$ echo The total is $100.00
The total is 00.00
```

প্রথম উদাহরণে আমি অনেকগুলো স্পেস দিয়েছি ঠিকই কিন্তু echo সেগুলো ধর্তব্যে আনেনি। দ্বিতীয়টিতে আমি ১০০ডলার লিখতে চেয়েছিলাম কিন্তু শুরুতে '\$' থাকায় শেল '\$1' কে ভেরিয়েবল হিসেবে বিবেচনা করেছে এবং তার কোনো মান শেলের জানা নেই বলে কিছুই বসায়নি। এই সমস্যা কিছুই এক্সপ্যানসনের ফল। এমন অবস্থা আসতে পারে যখন আমরা এক্সপ্যানসন চাই না। তখন আমরা তা বন্ধ করতে পারি ক্যাটিং(quoting) এর মাধ্যমে। ক্যাটিং এর জন্য দুটি চিহ্ন ব্যবহৃত হয়। সিঙ্গেল কোট(') ও ডবল কোট("")।

## ডবল কোট

যখন আপনি কোনো আর্গুমেন্টকে ডবলকোটে("") দিয়ে আবদ্ধ করবেন শেলে যেসব চিহ্নের বিশেষ কোনো অর্থবহন করে সেগুলো তাদের অর্থ হারাবে অর্থাৎ সাধারণ অক্ষরের মত ব্যবহৃত হবে। শুধুমাত্র "\$", "\", এবং "\"" এর ব্যতিক্রম। আমরা বরং একটা ছক থেকে দেখে নিই কোন কোন শেলফিচারগুলো ডবল কোটে নিষ্ক্রিয় থাকবে আর কোনগুলো সক্রিয়:

ফিচার	সক্রিয়তা/ নিষ্ক্রিয়তা	মন্তব্য
ওয়ার্ড- স্প্লিটিং	নিষ্ক্রিয়	আর্গুমেন্ট হিসেবে দেয়া শব্দগুলো স্পেস দিয়ে আলাদা হলে তাদের আলাদা আলাদা গন্য করা হত। ডবল কোটের মধ্যে থাকা স্পেসগুলোর জন্য তেমন হবে না।
পাথনেম এক্সপ্যানসন	নিষ্ক্রিয়	ওয়াইল্ডকার্ড ব্যবহার করে আমরা যে পাথনেম এক্সপ্যানসন করি তা নিষ্ক্রিয় থাকবে।
টিন্ডে এক্সপ্যানসন	নিষ্ক্রিয়	টিন্ডে চিহ্ন(~) আমরা হোমের বদলে ব্যবহার করতাম, এটি নিষ্ক্রিয় থাকবে।
ব্রেস এক্সপ্যানসন	নিষ্ক্রিয়	ব্রেস এক্সপ্যানসন নিষ্ক্রিয় থাকবে।
প্যারামিটার এক্সপ্যানসন	সক্রিয়	\$ ব্যবহার করে প্যারামিটার এক্সপ্যানসন করতে হয় তাই এটি সক্রিয় থাকবে।
গাণিতিক এক্সপ্যানসন	সক্রিয়	গাণিতিক এক্সপ্যানসনও \$ এর উপর নির্ভরশীল বলে এটিও সক্রিয় থাকবে।
কমান্ড সাবস্টিটিউশন	সক্রিয়	এটিও \$ এর উপর নির্ভরশীল তাই এটিও কাজ করবে।

এবার দেখা যাক কেন ও কখন ডবল কোট ব্যবহার করবো। আপনি ফাইল ম্যানেজার দিয়ে হোমে যান এবং দেখুন **Untitled Folder** নামে কোনো ফোল্ডার আছে কিনা। না থাকলে একটা নতুন ফোল্ডার তৈরি করুন, কোনো নাম না দিলে এই নামটিই ব্যবহার করবে। এবার টার্মিনালে লিখুন:

```
me@howtocode-pc:~$ ls -l Untitled Folder
ls: cannot access Untitled: No such file or directory
ls: cannot access Folder: No such file or directory
me@howtocode-pc:~$ mv "Untitled Folder" Untitled_Folder
```

আমরা কেবলই **Untitled Folder** ফোল্ডারটি তৈরি করেছি। কিন্তু আমরা যখন কমান্ডলাইনে `ls -l` কমান্ডের সাথে আর্গুমেন্ট হিসেবে ফোল্ডারটির নাম দিয়েছি মাঝখানে স্পেস থাকার কারণে ওয়ার্ড-স্প্লিটিং ঘটেছে। ফলে `Untitled` ও `Folder` নামে দুটো আর্গুমেন্ট তৈরি হয়েছে এবং কোনোটাই পাওয়া যায়নি।

লিনাক্স টাউশনে যারা কমান্ডলাইনে কাজ করেন তারা ফোল্ডারগুলোর নামের মধ্যে স্পেসের বদলে আন্ডারস্কোর(\_) ক্যারেক্টার ব্যবহার করেন দ্রুত কাজ করার সুবিধার্থে। আমরা তাই `Untitled Folder` থেকে নাম পরিবর্তন করে `Untitled_Folder` করেছি যেন বারবার এই সমস্যা পড়তে না হয়।

আমরা জেনেছি প্যারামিটার এক্সপ্যানসন, গাণিতিক এক্সপ্যানসন ও কমান্ড সাবস্টিটিউশন কাজ করবে। একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ echo "$USER $((2+2)) $(cal)"
me 4      September 2014
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
  7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30
```

আমরা **echo** কমান্ডের আর্গুমেন্ট হিসেবে `"$USER $((2+2)) $(cal)"` ব্যবহার করেছি। যার প্রথমে ছিল `USER` ভেরিয়েবল যার সামনে `$` ব্যবহার করে `$USER` লিখেছি প্যারামিটার এক্সপ্যানসন এর জন্য। ফলে আউটপুটের প্রথম লাইনে আমরা 'me' দেখতে পাচ্ছি যেটা আমার ইউজারনেম। এরপর আমরা একটা ম্যাথ এক্সপ্যানসন দিয়েছি: `$((2+2))`। যার উত্তর 4 সেটিও প্রিন্ট করেছে 'me' এর পরেই। তারপর আমরা **cal** কমান্ডের সাবস্টিটিউশন করেছি যা তারপরেই প্রিন্ট হয়েছে।

আমরা আগের লেসনে একটি উদাহরণ দেখেছিলাম:

```
me@howtocode-pc:~$ echo শেষ শব্দটি নেখার আগে আমি অনেকগুলো স্পেস দেবো।
শেষ শব্দটি নেখার আগে আমি অনেকগুলো স্পেস দেবো।
```

দেখুন, আমাদের অতিরিক্ত স্পেস কিন্তু দেখায়নি। কারণটা কি? কারণটা হল প্রথমে শেল যখন সম্পূর্ণ কমান্ডটি আমাদের কাছে পেলো। সে সবগুলো স্পেস কে ভেবে নিল আর্গুমেন্ট গুলো আলাদা করার উপায়। যাকে ডিলিমিটারস বলে। অতএব তার কাছে 'শেষ', 'শব্দটি', 'লেখার'... এরকম সব শব্দগুলো আলাদা আলাদা আর্গুমেন্ট হয়ে গেলো। শেলের এই ফিচারকে ওয়ার্ড-স্প্লিটিং বলে। শেল শব্দগুলোকে শুধু, স্পেসগুলো বাদে আলাদা আলাদা



আর্গুমেন্ট হিসেবে **echo** কমান্ডের কাছে পাঠালো। আর **echo** তার আর্গুমেন্টগুলো প্রিন্ট করার সময় তাদের মাঝে স্পেস দেয়। এবার আমরা যদি সম্পূর্ণ আর্গুমেন্টটাকে ডবল কোটে আবদ্ধ করে দেই তাহলে কিন্তু যেমন লিখেছি তেমনই দেখাবে:

```
me@howtocode-pc:~$ echo "শেষ শব্দটি নেখার আগে আমি অনেকগুলো স্পেস দেবো। "
```

```
শেষ শব্দটি নেখার আগে আমি অনেকগুলো স্পেস দেবো।
```

## সিঙ্গেল কোট

আমরা ডবল কোটের ক্ষেত্রে দেখেছি কিছু কিছু এক্সপ্যানসন সক্রিয় থাকে। কিন্তু আমরা যদি সকলরকমের এক্সপ্যানসন নিষ্ক্রিয় করতে চাই তাহলে সিঙ্গেল কোট(') ব্যবহার করতে পারি। আমরা একটি **echo** কমান্ডের আর্গুমেন্টকে আলাদা আলাদাভাবে স্বাভাবিকভাবে এবং ডবল কোট ও সিঙ্গেল কোটে আবদ্ধ করলে পার্থক্যটি বুঝতে পারবো:

```
me@howtocode-pc:~$ echo text ~/.txt {a,b} $(echo foo) $((2+2)) $USER
text /home/me/ls-output.txt /home/me/ls.txt a b foo 4 me
me@howtocode-pc:~$ echo "text ~/.txt {a,b} $(echo foo) $((2+2)) $USER"
text ~/.txt {a,b} foo 4 me
me@howtocode-pc:~$ echo 'text ~/.txt {a,b} $(echo foo) $((2+2)) $USER'
text ~/.txt {a,b} $(echo foo) $((2+2)) $USER
```

প্রথমে আমরা স্বাভাবিকভাবে `me@howtocode-pc:~$ echo text ~/.txt {a,b} $(echo foo) $((2+2)) $USER` কমান্ডটি দিয়েছি। এখানে প্রথমে পাথনেম এক্সপ্যানসন, তারপর ব্রেস এক্সপ্যানসন, তারপর কমান্ড সাবস্টিটিউশন, গাণিতিক এক্সপ্যানসন ও প্যারামিটার এক্সপ্যানসন সবই হয়েছে। তারপর আমরা আর্গুমেন্টটিকে ডবল কোটে আবদ্ধ করে দিয়েছি। ফলে পাথনেম এক্সপ্যানসন ও ব্রেস এক্সপ্যানসন ঘটেনি কিন্তু কমান্ড সাবস্টিটিউশন, গাণিতিক এক্সপ্যানসন ও প্যারামিটার এক্সপ্যানসন ঘটেছে। শেষ উদাহরণে আমরা আর্গুমেন্ট সিঙ্গেল কোটে আবদ্ধ করেছি ফলে কোনো এক্সপ্যানসনই হয়নি।

## ক্যারেটার স্কেইপিং

আমরা জেনেছি \$, !, \*, & এই চিহ্নগুলো বিশেষ অর্থবহ শেলের কাছে। এমনকি স্পেস ব্যবহৃত হয় ওয়ার্ড-স্প্লিটিং এর জন্য। \$ চিহ্নটি ব্যবহৃত হয় নানারকমের এক্সপ্যানসনের জন্য। এখন আমরা কখনো কমান্ডের মধ্যে যদি চাই কিছু এক্সপ্যানসন কাজ করবে ও কিছু কাজ করবে না তাহলে আমরা ক্যারেটার স্কেইপিং এর সাহায্য নিতে পারি। এজন্য আমরা ব্যবহার করবো ব্যাকস্ল্যাশ() চিহ্নটি। উল্লেখ্য স্বাভাবিক অবস্থায় ও ডবল কোটের মধ্যে ক্যারেটার স্কেইপ করা যায়। সিঙ্গেল কোটের ভিতর এটি সম্ভব না। একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ echo "The balance for user $USER is: \"$5.00"
The balance for user me is: $5.00
```

উপরের **echo** কমান্ডের আর্গুমেন্ট ডবল কোটে আবদ্ধ ছিল। আমরা জানি, ডবল কোটের মধ্যে ডেরিয়েবল বা প্যারামিটার এক্সপ্যানসন কাজ করে। ফলে আউটপুটে \$USER কে এক্সপ্যান্ড করে বর্তমান ইউজার অর্থাৎ 'me' দেখাবে। একই ভাবে \$5.00 এর \$5কেও ডেরিয়েবল হিসেবে এক্সপ্যান্ড করত এবং 5 এর জন্য কোনো মান নির্দেশ করা নেই বলে '.00' দেখাত কিন্তু আমরা '\' চিহ্ন দিয়ে ওই এক্সপ্যানসন স্কেইপ করেছি ফলে আউটপুটে \$5.00ই দেখাচ্ছে।

## অধ্যায় - পাঁচ

# কীবোর্ড ট্রিক্স

আপনার হয়ত মনে হতে পারে কমান্ডলাইন ইউজাররা প্রচুর টাইপ করতে পছন্দ করে। বাস্তবতা কিন্তু উল্টো। যা সময় নষ্ট করে এমনকি তা যদি টাইপ করাও হয় তবে তা ঘৃণা করে কমান্ডলাইন ইউজাররা। চেষ্টা করা হয় যত কম টাইপ করে, যত কম প্রচেষ্টা ও পুনরাবৃত্তিতে কিছু করা যায়। এজন্য লক্ষ্য করে দেখবেন, অধিকাংশ কমান্ডগুলোর নাম বেশ ছোট। যেমন: cp, mv, rm, pwd...

এই অধ্যায়ে আমরা কিছু শেল ফিচার ও কিছু কীবোর্ডের ব্যবহারের উপর জোর দেবো যা আমাদের কম কষ্টে ভালোভাবে কাজ করার সুযোগ দেবে।

- **কমান্ডলাইন এডিটিং:** কমান্ডলাইন এডিটিং এর শর্টকাটসমূহ।
- **কমপ্লিশন:** কমপ্লিশনের ব্যবহার।
- **কমান্ড হিস্ট্রি:** কমান্ড হিস্ট্রির ব্যবহার।

## কমান্ডলাইন এডিটিং

Readline নামের একটা লাইব্রেরির মাধ্যমে bash কমান্ডলাইন এডিটিং এর সুযোগ দেয়। কমান্ডলাইন এডিটিং এর কিছু নিদর্শন আমরা আগেও দেখেছি। যেমন আমরা এ্যারো কি ব্যবহার করে কমান্ডের লেখার মধ্যে ডানে-বামে সরতে পারি। এমন না যে সমস্ত এডিটিং ফিচার আপনার জানতেই হবে। তবে কোন কোনটা আপনার কাজে লাগতেও পারে। দেখে নেয়া ভালো।

### কার্সর মুভমেন্ট

এখানে কার্সর মুভমেন্ট এর কিছু কীবোর্ড শর্টকাট দেয়া হল:

কী	কার্যকারিতা
Ctrl-a	কার্সরকে লাইনের প্রথমে নিয়ে যাবে।
Ctrl-e	কার্সরকে লাইনের শেষে নিয়ে যাবে।
Ctrl-f	কার্সরকে এক অক্ষর সামনে নিয়ে যাবে। রাইট এ্যারো কী চাপলেও একই ব্যাপার ঘটবে।
Ctrl-b	কার্সরকে এক অক্ষর পিছনে নিয়ে যাবে। লেফট এ্যারো কী চাপলেও একই ব্যাপার ঘটবে।
Alt-f	কার্সরকে এক শব্দ সামনে নিয়ে যাবে।
Alt-b	কার্সরকে এক শব্দ পিছনে নিয়ে যাবে।
Ctrl-l	স্ক্রীনের সবকিছু মুছে ফেলে উপরের বামপাশে একটা প্রম্পট ও কার্সর হাজির হবে। <b>clear</b> কমান্ড দিলেও একই ব্যাপার ঘটবে।

### এডিটিং শর্টকাট

এবার কিছু টেক্সট এডিটিং শর্টকাট দেখে নেয়া যাক:

কী	কার্যকারিতা
Ctrl-d	কার্সরের অবস্থানের একটি অক্ষর মুছে ফেলবে।
Ctrl-t	কার্সরের অবস্থানের ও তার পরবর্তী অক্ষরের মধ্যে জায়গার অদলবদল ঘটবে।
Alt-t	কার্সরের অবস্থানের ও তার পরবর্তী শব্দের মধ্যে জায়গার অদলবদল ঘটবে।
Alt-l	কার্সরের অবস্থান থেকে লাইনের শেষপর্যন্ত সমস্ত লেখা ছোটহাতের অক্ষরে বদলে যাবে।
Alt-u	কার্সরের অবস্থান থেকে লাইনের শেষপর্যন্ত সমস্ত লেখা বড়হাতের অক্ষরে বদলে যাবে।

## কাট এবং পেস্ট

আমরা যাকে কাট এবং পেস্ট বলি readline লাইব্রেরিটির ডকুমেন্টেশনে তাকে কিলিং(killing) ও ইয়াকিং(Yanking) বলে। যে লেখাগুলো কাট বা কিল করা হয় তা কিল-রিং(kill-ring) নামের একটা জায়গায় রাখা হয় যার পোশাকি নাম বাফার(buffer)। এখানে কিছু কাট এবং পেস্ট কমান্ড দেয়া হল:

কী	কার্যকারিতা
Ctrl-k	কার্সরের অবস্থান থেকে লাইনের শেষ পর্যন্ত কাট বা কিল করবে।
Ctrl-u	কার্সরের অবস্থান থেকে লাইনের শুরু পর্যন্ত কাট বা কিল করবে।
Alt-d	কার্সরের অবস্থান থেকে শব্দের শেষ পর্যন্ত কাট বা কিল করবে।
Alt-Backspace	কার্সরের অবস্থান থেকে শব্দের শুরু পর্যন্ত কাট বা কিল করবে।
Ctrl-y	কিল-রিং থেকে লেখা কপি করবে এবং কার্সরের অবস্থানে পেস্ট করবে।

## কমপ্লিশন

শেলের একটা খুব কার্যকারী ক্ষমতা হচ্ছে কমপ্লিশন(Completion)। আপনি কিছুটা লিখে ট্যাব চাপলে কমান্ড বা আর্গুমেন্ট বা ফাইলপাথ শেল নিজেই বাকিটুকু লিখে দেয়। উদাহরণ দেখা যাক বরং:

```
me@howtocode-pc:~$ cd Music
me@howtocode-pc:~/Music$ ls
Bob Dylan/                               Soulful-Voice-Arjit-Singh-128Kbps-2014(
Music/                                     Sufi/
Pete Seeger - The Essential Pete Seeger (2005)/ Veer/
Ringtones/                               জন এর গান/
```

আমরা প্রথমে Music ফোল্ডারে ঢুকেছি। তারপর ls লিখে স্পেস দিয়েছি। ls কমান্ডটি আর্গুমেন্ট হিসেবে ফাইলপাথ নেয়। সেজন্য এরপর আমরা যখন ট্যাব চাপলাম ওই ফোল্ডারে থাকা সকল কিছুর লিস্ট দেখিয়েছে। এখন আমরা যদি ls B পর্যন্ত লিখে ট্যাব দিই, কি হয় দেখুন:

```
me@howtocode-pc:~/Music$ ls Bob\ Dylan/
```

ls B লিখে ট্যাব চাপার ফলে ওই ফোল্ডারে B দিয়ে শুরু হওয়া একমাত্র ফোল্ডারটিকে শেল আর্গুমেন্ট হিসেবে বেছে নিয়েছে। কিন্তু লক্ষ্য করুন, এখানে 'S' দিয়ে শুরু দুটো ফোল্ডার আছে। সুতরাং 'ls S' পর্যন্ত লিখে কোনটা বেছে নেবে? উত্তর হচ্ছে কোনোটাই বেছে নেবে না। একবার ট্যাব চাপলে কিছুই হবে না আর দুবার চাপলে 'S' দিয়ে শুরু সবকিছুর নাম দেখাবে এভাবে:

```
me@howtocode-pc:~/Music$ ls S
Soulful-Voice-Arjit-Singh-128Kbps-2014(Songs.PK)/ Sufi/
me@howtocode-pc:~/Music$ ls S
```

পরের লাইনে আমরা যদি আবার ls S ফিরে এসেছে। এবার যদি ls Su লিখে ট্যাব চাপি তাহলে শেল নিশ্চিতভাবে বুঝে যাবে আমরা Sufi বোঝাতে চেয়েছি।

কমপ্লিশন কমান্ডের নাম, পাথনেম ছাড়াও ভেরিয়েবলের ক্ষেত্রেও কাজ করে।

## কমান্ড হিস্ট্রি

আমরা যেসব কমান্ড শেলে দিই, শেল সেগুলোকে মনে রাখে। সাধারণত শেষ ৫০০ কমান্ড শেল সংরক্ষণ করে `.bash_history` নামে একটি বিশেষ ফাইলে। আমরা আপার এ্যারো কী চেপে একটা একটা করে কমান্ড দেখতে পারি। বা **history** কমান্ড দিয়ে সম্পূর্ণ লিস্টটি দেখতে পারি। তবে এই দীর্ঘ লিস্ট কমান্ডলাইনে স্বাভাবিকভাবে দেখা বেশ ঝামেলার। ভালো উপায় হল এর আউটপুট কে **less** কমান্ডে রিডিরেক্ট করে দেখা:

```
me@howtocode-pc:~$ history | less
```

আমরা চাইলে হিস্ট্রির মধ্যে **grep** দিয়ে কোনো প্যাটার্ন খুঁজে দেখতে পারি। যেমন আমরা যদি চাই সেইসব কমান্ড দেখবো যার মধ্যে **/usr/bin** আছে তবে লিখবো:

```
me@howtocode-pc:~$ history | grep /usr/bin
```

এর মধ্যে ৫৯৯ লাইনে আমি এই কমান্ডটি পেয়েছি:

**599 ls -l /usr/bin/ > ls-output.txt**

আমরা কোনো কমান্ডকে তার লাইন নম্বর দিয়েও প্রম্পটে আনতে পারি যদি চাই। যেমন উপরের এই কমান্ডটিকে ডাকতে পারি এভাবে:

```
me@howtocode-pc:~$ !599
ls -l /usr/bin/ > ls-output.txt
```

**bash** এর একটা অসাধারণ ফিচার হলো ইন্টারেক্টিভ রিডার্স ইনক্রিমেন্টাল হিস্ট্রি সার্চ। আসুন দেখা যাক। মনে করুন, স্বাভাবিকভাবে এই আমাদের প্রম্পট:

```
me@howtocode-pc:~$
```

এবার **Ctrl-r** চাপুন। দেখবেন প্রম্পট এমন হয়ে গেছে:

```
(reverse-i-search)`:
```

এবার আপনি লিখতে থাকুন **/usr/bin**। দেখবেন প্রতিটা অক্ষর লেখার সাথে সাথে শেল তারসাথে মিলিয়ে শেষ যে কমান্ডটি এক্সিকিউট করেছে দেখাবে। আপনি এখান থেকে কপি করতে পারবেন বা এন্টার চাপলে কমান্ডটি আবার এক্সিকিউট হবে। এবার দেখা যাক হিস্ট্রি ব্যবহারের জন্য কিছু কীবোর্ড শর্টকাট:

কী	কার্যকারিতা
Ctrl-p	পূর্ববর্তী কমান্ড দেখাবে। আপ এ্যারো কী চাপলেও একই ব্যাপার ঘটবে।
Ctrl-n	পরবর্তী কমান্ড দেখাবে। ডাউন এ্যারো কী চাপলেও একই ব্যাপার ঘটবে।
Alt-<	হিস্ট্রি লিস্টের প্রথম কমান্ডে যাবে।
Alt->	হিস্ট্রি লিস্টের শেষ অবস্থান অর্থাৎ বর্তমান প্রম্পটে ফিরে আসবে।
Ctrl-r	রিভার্স ইনক্রিমেন্টাল সার্চ প্রম্পট আসবে।
Alt-p	ননইনক্রিমেন্টাল রিভার্স সার্চ। অর্থাৎ আপনাকে সার্চ টার্ম লিখে এন্টার দিতে হবে এবং আপনি ওই সার্চ টার্মের সাথে মেলে এমন শেষ কমান্ডটি পাবেন।
Alt-n	ফরওয়ার্ড সার্চ, ননইনক্রিমেন্টাল।
Ctrl-o	হিস্ট্রির বর্তমান কমান্ডটি এক্সিকিউট করবে ও পরের কমান্ডে চলে যাবে। এটা বেশ কার্যকর যদি আপনি আগে কখনো পরপর দেয়া কিছু কমান্ড আবার একইভাবে দিতে চান।

## হিস্ট্রি এক্সপ্যানসন

শেল হিস্ট্রি এক্সপ্যানসন নামের এক বিশেষ ধরনের এক্সপ্যানসনের সুযোগ দেয়। যা শুরু হয় "!" চিহ্ন দিয়ে। কিছু হিস্ট্রি এক্সপ্যানসন কমান্ড দেখা যাক:

কমান্ড কাঠামো	কার্যকারিতা
!!	সর্বশেষ ব্যবহৃত কমান্ডটি দেখাবে। আপ এ্যারো কী চাপলেও একই কাজ হবে।
!number	'number' এর জায়গায় দেয়া সংখ্যা অনুযায়ী তত নম্বর কমান্ডটি খুঁজে বের করবে।
!string	শেষ যে কমান্ডটি 'string' এর জায়গায় লেখা স্ট্রিং দিয়ে শুরু হয়েছে সেটি খুঁজে বের করবে।
!?string	শেষ যে কমান্ডটির মধ্যে 'string' এর জায়গায় লেখা স্ট্রিং আছে সেটি খুঁজে বের করবে।



## অধ্যায় - ছয়

# পারমিশন

আমরা যদি লিস্ট করতে বসি এমন ১০টা কারনের যার জন্য উইন্ডোজ থেকে আসা নতুন লিনাক্স ইউজার ঘাবড়ে যান তার মধ্যে পারমিশন একটা হবেই। একজন শিক্ষানবীশ ওয়েবডেভেলপার যে wampp এ কাজ করে অভ্যস্ত সে হঠাৎই দেখতে পায় পারমিশনের কারনে তার সার্ভার ঠিকমত কাজ করছে না। বা সিস্টেমের কোনো ম্যানিপুলেশনমাত্রই সুপারইউজার এ্যাবিলিটি দরকার হয়। উইন্ডোজেও এডমিনিস্ট্রেটিভ প্রিভিলেজ চায় অবশ্য কিন্তু next এবং ok চাপার অভ্যেসবশত আমরা ok চাপতে কসুর করি না।

লিনাক্স ডিস্ট্রিবিউশনগুলো ইউনিক্সসদৃশ অপারেটিং সিস্টেম(আপনি হয়ত জেনে থাকবেন, লিনাক্স অপারেটিং সিস্টেম না বরং একটি কার্নেল। কার্নেল অপারেটিং সিস্টেমের একটি অতি গুরুত্বপূর্ণ অংশ তবুও পুরো অপারেটিং সিস্টেম না। লিনাক্স কার্নেল ব্যবহার করে তৈরি করা অপারেটিং সিস্টেমকে লিনাক্স ডিস্ট্রিবিউশন বলে।) এবং স্বাভাবিকভাবেই ইউনিক্সের মত অনেক কাজ একসাথে করার(Multitasking) ও অনেক ইউজারকে একসাথে কাজ করার(Multiuser) সুযোগ দেয়।

তাই স্বাভাবিকভাবেই একটা কম্পিউটারের একাধিক ইউজার থাকতে পারে। তাদের প্রাইভেসির দরকার হয়। যেন তারা অন্যের ফাইলপত্রে অনিধকার অনুপ্রবেশ না করতে পারে এবং তাদের তথ্যও সুরক্ষিত থাকে। এর জন্য দরকার ফাইল পারমিশন। আবার একটা সিস্টেমে হাজার হাজার ইউজার থাকতে পারে। আমরা নিশ্চয়ই তখন চাইবো না কেউ সিস্টেমের কোনো অংশ বদলে তাকে ক্ষতিগ্রস্ত করুক। তাই ব্যবহারকারীদের মধ্যেও সাধারণ ও সুপারইউজার শ্রেণী আছে।

- **ওনার, গ্রুপ এবং অন্যান্য:** ফাইল ওনার সম্পর্কিত ধারণা।
- **এক্সেস রাইট:** ফাইল এক্সেস রাইট সম্পর্কিত ধারণা।
- **ফাইল পারমিশন পরিবর্তন:** chmod এর ব্যবহার করে ফাইল পারমিশন পরিবর্তন।
- **ফাইল পারমিশন মাস্কিং:** umask এর ব্যবহার করে মাস্কিং।
- **বিশেষ পারমিশন:** বিশেষ পারমিশনসমূহের ধারণা।
- **ওনার ইউজার ও গ্রুপ পরিবর্তন:** chown ও chgrp এর ব্যবহার।
- **পরিচয় পরিবর্তন:** su এবং sudo এর ব্যবহার।
- **পাসওয়ার্ড পরিবর্তন:** passwd কমান্ড এর ব্যবহার।

## ওনার, গ্রুপ এবং অন্যান্য

আমুন ঝটপট `/etc/shadow` ফাইলটা দেখা যাক। প্রথমে **file** পরে **less** কমান্ড দিয়ে:

```
me@howtocode-pc:~$ file /etc/shadow
/etc/shadow: regular file, no read permission
me@howtocode-pc:~$ less /etc/shadow
/etc/shadow: Permission denied
```

আমরা দেখতে পেলাম, সাধারণ ইউজার হিসেবে আমাদের অধিকার নেই ফাইলটি পড়ার। লিনাক্স সিস্টেমে প্রত্যেক ফাইলের কোনো না কোনো মালিক বা ওনার(owner) আছে। এবং সেই ওনারই ঠিক করেন কারা কারা ফাইলটি দেখতে পারবেন, এডিট করতে পারবেন বা ব্যবহার করতে পারবেন। সেই ওনার হয়ত কোনো গ্রুপের অন্তর্ভুক্ত যেখানে আরো ইউজার আছে। তাকে এটাও ঠিক করে দিতে হবে যে গ্রুপের অন্য সদস্যরা ফাইলটিতে কী ধরনের কাজ করতে পারবে। এবং এসবের বাইরেও যেকোনো কিভাবে ফাইলটি ব্যবহার করতে পারবে। এই বাকি সব ইউজারদের ইউনিক্স সিস্টেমে world বলা হয়। আপনি হয়ত বুঝতে পারছেন আপনার ক্ষমতার অনেকটাই নির্ভর করে আপনি কোনধরনের ইউজার এবং কোন কোন গ্রুপের অন্তর্ভুক্ত। এটা আপনি জানতে পারেন **id** কমান্ড দিয়ে:

```
me@howtocode-pc:~$ id
uid=1001(me) gid=1001(me) groups=1001(me),4(adm),27(sudo),44(video),99(portal),108(lpadmin)
```

একজন ইউজার তৈরি হওয়ার সময়ই তার একটা uid বা user id দেয়া হয়। উবুন্টুতে সাধারণ ইউজারের uid শুরু হয় 1000 থেকে তাই আমার ইউজারের uid 1001। তাছাড়া ইউজারকে একটি gid বা group id দিয়ে তার নিজের নামেই একটি প্রাথমিক গ্রুপে যুক্ত করা হয়। সে যতগুলো গ্রুপে আছে তাও আমরা দেখতে পাই এখানে।

একটা কমন প্রশ্ন হচ্ছে এই তথ্যগুলো কোথায় থাকে? **/etc/passwd** ফাইলে ইউজার ও **/etc/group** ফাইলে গ্রুপ সংক্রান্ত তথ্য থাকে। এনক্রিপ্টেড অবস্থায় পাসওয়ার্ড থাকে **/etc/shadow** ফাইলে।

## এক্সেস রাইট

একটা ফাইল বা ডিরেক্টরিতে কে কতটুকু কি করতে পারবে তা এক্সেস রাইট(access right) দ্বারা নির্ধারিত হয়। এক্সেস রাইট তিনরকমের হয়। রিড এক্সেস(read access), রাইট এক্সেস(write access) ও এক্সিকিউশন এক্সেস(execution access)। যার রিড এক্সেস আছে সে ফাইলটি পড়তে পারবে। যার রাইট এক্সেস আছে সে সেই তথ্য পরিবর্তন করতে পারবে এবং যার এক্সিকিউশন এক্সেস আছে সে পারবে সেটিকে রান করাতে। একজন একাধিক বা সবগুলো এক্সেসই পেতে পারে। এবার আমরা উদাহরণে যাই। প্রথমে আমরা **foo.txt** নামে একটি ফাইল তৈরি করে তার পারমিশনগুলো দেখাবো। `ls` কমান্ড দিয়ে:

```
me@howtocode-pc:~$ ls -l foo.txt
-rw-rw-r-- 1 me me 0 Sep 23 18:02 foo.txt
```

'-' চিহ্নসহ প্রথম যে ১০ অক্ষর(এখানে -rw-rw-r--), এটিই ফাইল এক্সেস পারমিশন সম্পর্কিত তথ্য দেয়। এর প্রত্যেক অক্ষরের নির্দিষ্ট অর্থ আছে।

প্রথম অক্ষর এখানে যেটা '-' চিহ্ন এটা ফাইল টাইপ(file type) অর্থাৎ ফাইলটি কী ধরনের ফাইল তা নির্দেশ করে। এখানে '-' চিহ্ন দিয়ে বোঝানো হয়েছে এটি একটি রেগুলার বা সাধারণ ফাইল। আরো কয়েকরকম ফাইল টাইপ আছে যা বিভিন্ন অক্ষর দিয়ে প্রকাশ করা হয়। এই অক্ষরগুলোকে বলা হয় এটিবিউট। আসুন অন্য ফাইল টাইপ এটিবিউট এবং তার অর্থ দেখে নেয়া যাক:

এটিবিউট	ফাইল টাইপ
-	সাধারণ ফাইল বা রেগুলার ফাইল।
d	ডিরেক্টরি, যাকে আমরা ফোল্ডারও বলে থাকি।
l	সিম্বোলিক লিঙ্ক।
c	ক্যারেটার স্পেশাল ফাইল। এমনসব ডিভাইস যারা বাইট পর্যায়ে ডাটা আদানপ্রদান করে থাকে। যেমন টার্মিনাল বা মডেম।
b	ব্লক স্পেশাল ফাইল। এমনসব ডিভাইস যারা ডাটা ব্লক হিসেবে ট্রান্সফার করে। যেমন হার্ডড্রাইভ বা সিডি।

বাকি নয়টি অক্ষর ফাইল মোড নির্দেশ করে। প্রতি তিনটি করে অক্ষর নিয়ে আমরা মোট তিন ভাগে ভাগ করতে পারি। প্রথম ভাগ ওনার এর, দ্বিতীয় ভাগ গ্রুপ আর শেষ ভাগ ওয়ার্ল্ড এর। এরকম:

ওনার	গ্রুপ	ওয়ার্ল্ড
rwX	rwX	rwX

এখানে মাত্র তিনটি এটিবিউট:

- **r**: রিড এক্সেস।
- **w**: রাইট এক্সেস।

- **x**: এক্সিকিউশন এক্সেস।

এবার দেখে নেয়া যাক কোনধরনের এক্সেস থাকলে আপনি কি কি করতে পারবেন:

এটিবিউট	ফাইল	ডিরেক্টরি
r	ফাইল খুলতে ও পড়তে দেয়।	ডিরেক্টরির কন্টেন্টের লিস্ট করতে দেয় যদি সাথে এক্সিকিউশন এক্সেসও থাকে।
w	ফাইলে লিখতে, বা তথ্য মুছে দিতে দেবে। কিন্তু ফাইলটিকে মুছতে বা তার নাম পরিবর্তন করতে দেবে। ফাইল মোছা বা তার নাম পরিবর্তনের ক্ষমতা ডিরেক্টরির পারমিশনের ওপর নির্ভরশীল।	ডিরেক্টরির মধ্যে নতুন ফাইল তৈরী করা, মুছে ফেলা বা নাম পরিবর্তনের সুযোগ দেয় সাথে এক্সিকিউশন এক্সেস থাকলে।
x	ফাইলকে প্রোগ্রাম হিসেবে বিবেচনা করবে ও এক্সিকিউট করবে। তবে প্রোগ্রাম ফাইলটি স্ক্রিপ্টিং ল্যাঙ্গুয়েজ(যেমন: পাইথন, শেলস্ক্রিপ্ট) এ লেখা হলে রিড এক্সেসও থাকতে হবে।	ডিরেক্টরিতে ঢুকতে দেবে।

কারো কোনো পারমিশন থাকলে তার নির্দিষ্ট জায়গায় সেই এটিবিউটের অক্ষর থাকবে। না হলে '-' চিহ্ন। আসুন কিছু উদাহরণ দেখে নেয়া যাক:

এটিবিউট	অর্থ
-rwx-----	একটা সাধারণ ফাইল যার ওনারের সকল পারমিশন আছে কিন্তু গ্রুপ ও ওয়ার্ল্ডের কোনো পারমিশন নেই।
-rw-----	একটা সাধারণ ফাইল যার ওনারের শুধু রিড ও রাইট পারমিশন আছে আর কারো কোনো পারমিশন নেই।
-rw-r--r--	একটা সাধারণ ফাইল যার ওনারের রিড ও রাইট এবং গ্রুপ ও ওয়ার্ল্ডের রিড পারমিশন আছে।
-rwxr-xr-x	একটা সাধারণ ফাইল যার ওনারের সকল পারমিশন আছে এবং গ্রুপ ও ওয়ার্ল্ডের রিড ও এক্সিকিউশন পারমিশন আছে।
-rw-rw----	একটা সাধারণ ফাইল যার ওনার ও গ্রুপের রিড ও রাইট পারমিশন আছে, ওয়ার্ল্ডের কোনো পারমিশন নেই।
lrwxrwxrwx	একটি সিম্বোলিক লিঙ্ক ফাইল যার সবার সকল পারমিশন আছে। সিম্বোলিক লিঙ্কের সবসময় একই পারমিশন থাকে এবং এটি ডামি বা নকল। ফাইলটি যে ফাইলের লিঙ্ক, সেই ফাইলটির পারমিশন আসলে কার্যকর থাকে।
drwxrwx---	একটি ডিরেক্টরি যার ওনার ও গ্রুপের সকল পারমিশন আছে কিন্তু ওয়ার্ল্ডের কোনো পারমিশন নেই।
drwxr-x---	একটি ডিরেক্টরি যার ওনারের সকল পারমিশন আছে এবং গ্রুপের রিড ও এক্সিকিউশন পারমিশন আছে কিন্তু ওয়ার্ল্ডের কোনো পারমিশন নেই।

## ফাইল পারমিশন পরিবর্তন

বিভিন্ন প্রয়োজনে আপনাকে ফাইল পারমিশনে পরিবর্তন আনতে হতে পারে। আপনি সেটা **chmod** কমান্ড দিয়ে সহজেই করতে পারেন। একটা ফাইলের পারমিশন শুধু তার ওনার ও এবং সুপারইউজার চেঞ্জ করতে পারে।

**chmod** দুটি পদ্ধতিতে পারমিশন পরিবর্তন করার সুযোগ দেয়। অকটাল এবং সাংকেতিক পদ্ধতি।

### অকটাল পদ্ধতি

অকটাল একধরনের গণনাপদ্ধতি। আমরা যেমন সাধারণ জীবনে দশভিত্তিক গণনা পদ্ধতি ব্যবহার করি, আবার যেমন দুই-ভিত্তিক বাইনারির কথা শুনেছি, অকটাল তেমনি ৮ ভিত্তিক। অর্থাৎ এতে ৭ এর পর আট না হয়ে হয় ১০। সেই দশ অবশ্যই অকটাল এর ক্ষেত্রে ১০, দশমিকে তার মান ৮ই। পদ্ধতিটা এরকম:

০, ১, ২, ৩, ৪, ৫, ৬, ৭, ১০, ১১, ১২, ১৩, ১৪, ১৫, ১৬, ১৭, ২০...

আমরা অকটালের ০-৭ পর্যন্ত আটটি সংখ্যা ব্যবহার করে আটটি ফাইলমোড নির্ধারন করতে পারি এভাবে:

অকটাল	ফাইল মোড
0	---
1	--X
2	-W-
3	-WX
4	r--
5	r-X
6	rw-
7	rwX

এই মানগুলো ব্যবহার করে আমরা তিন সংখ্যার অকটাল নম্বর ব্যবহার করে ফাইলমোড চেঞ্জ করতে পারি। যার প্রথম সংখ্যা ওনার, দ্বিতীয় সংখ্যা গ্রুপ ও তৃতীয় সংখ্যা ওয়ার্ল্ড এর পারমিশন বোঝাবে একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ > foo.txt
me@howtocode-pc:~$ ls -l foo.txt
-rw-rw-r-- 1 nishadsingha portal 0 Sep 27 07:48 foo.txt
me@howtocode-pc:~$ chmod 600 foo.txt
me@howtocode-pc:~$ ls -l foo.txt
-rw----- 1 nishadsingha portal 0 Sep 27 07:48 foo.txt
```

উদাহরণে আমরা প্রথমে **foo.txt** নামে একটি ফাইল তৈরী করেছি। ফাইলটির পারমিশন দেখেছি `ls -l foo.txt` কমান্ড দিয়ে। এরপর **chmod** কমান্ড দিয়ে ফাইল পারমিশন চেঞ্জ করেছি। আমরা **chmod** এর জন্য দুটি আর্গুমেন্ট ব্যবহার করেছি। প্রথমে নতুন পারমিশনজ্ঞাপক সংখ্যা **600** এবং তারপর যে ফাইলটার পারমিশন পরিবর্তন করতে হবে সেটি। এবং আমরা তারপর আবার চেক করে দেখেছি পারমিশন পরিবর্তিত হয়েছে।

## সাংকেতিক পদ্ধতি

সাংকেতিক পদ্ধতি ফাইল পারমিশন পরিবর্তনের আরেকটি উপায়। আমাদের আগে চেনা রিড, রাইট, এবং এক্সিকিউশনের *r*, *w* ও *x* চিহ্নগুলোও এখানে কার্যকর। তাছাড়া আরও আছে *u*, *g*, *o* এবং *a*। আসুন এগুলোর অর্থ জেনে নেয়া যাক:

সংকেত	অর্থ
u	ইউজারের সংক্ষিপ্ত রূপ। এখানে ইউজার বলতে ফাইলের মালিক ইউজার।
g	গ্রুপ। ফাইলের মালিক গ্রুপ।
o	Other বা ওয়ার্ল্ড।
a	অল বা উপরের সবাই।

এছাড়াও তিনটি গাণিতিক চিহ্নও ব্যবহার করা হয়। এগুলো হল:

চিহ্ন	অর্থ
+	পারমিশন যোগ করা হবে।
-	পারমিশন তুলে নেয়া হবে।
=	যোগ-বিয়োগ প্রয়োজনমত করে একটি নির্দিষ্ট পারমিশন দেয়া হবে।

সাংকেতিক উপায়ে পারমিশন দিতে তিনটি জিনিস যথাক্রমে লিখতে হয়:

- প্রথমে লিখতে হয় কাকে পারমিশন দেয়া হচ্ছে। অর্থাৎ ইউজার হলে *u*, গ্রুপ ও ওয়ার্ল্ড হলে যথাক্রমে *g* ও *o* এবং সবাইকে পারমিশন দিতে *a*। তবে *a* এর ক্ষেত্রে কিছু না লিখলেও হয়।
- পারমিশনে কীভাবে পরিবর্তন আনা হবে। অর্থাৎ আমরা কি '+' চিহ্নের মাধ্যমে পারমিশন যোগ করবো? নাকি কোনো পারমিশন বিয়োগ করবো '-' চিহ্ন দিয়ে? অথবা চাইলে নির্দিষ্ট পারমিশন চেঞ্জ করতে পারি '=' চিহ্ন দিয়ে।
- কোন পারমিশনের পরিবর্তন ঘটাবো। এটা আমরা *r*, *w*, *x* দিয়ে বলতে পারি।

এবার সাংকেতিক পদ্ধতির কিছু উদাহরণ দেখা যাক:

সংকেত	অর্থ
u+x	মালিক ইউজারকে এক্সিকিউটের পারমিশন দেবে।
u-x	মালিক ইউজারের এক্সিকিউশন পারমিশন উঠিয়ে নেওয়া হবে।
+x	+x ও a+x একই কথা। অর্থাৎ সবাইকে এক্সিকিউশন পারমিশন দেয়া হলো।
o-rw	ওয়ার্ল্ডের রিড এবং রাইট পারমিশন উঠিয়ে নেওয়া হবে।
go=rw	গ্রুপ ও ওয়ার্ল্ডের শুধু রিড ও রাইট পারমিশন দেয়া হবে। এর কোনোটা আগে না থাকলে যোগ করা হবে এবং এক্সিকিউটেবল পারমিশন আগে থাকলে সেটা তুলে নেওয়া হবে।
u+x,go=rx	এখানে কমা দিয়ে আলাদা করা দুটো সংকেত একত্রে ব্যবহৃত হয়েছে। প্রথমটি দিয়ে ইউজারকে এক্সিকিউশন পারমিশন দেওয়া হবে, পরেরটি দিয়ে গ্রুপ ও ওয়ার্ল্ডের শুধু রিড ও এক্সিকিউশন পারমিশন দেয়া হবে।

## ফাইল পারমিশন মাস্কিং

মাস্ক বা মুখোশের সাথে আমরা সবাই পরিচিত। মুখোশের মূল কাজটা কি? মূল কাজ হলো মুখ ঢেকে রাখা। সব মুখোশই মুখ ঢাকে। কোনোটা পুরোপুরি কোনোটা আংশিক। **umask** কমান্ডটি ফাইল পারমিশনের ক্ষেত্রে এরকম মাস্কের কাজ করে। এটি দিয়ে আপনি অকটাল পদ্ধতির একটা মাস্ক পরিয়ে দিতে পারেন ফাইলকে। একটা ফাইল তৈরী হওয়ার সময় ডিফল্টভাবে যে পারমিশন পায়, এই মাস্ক পরলে তার খানিকটা ঢাকা পরবে অর্থাৎ বাদ যাবে। মাস্কিং কিন্তু পুরোনো ফাইল বা ডিরেক্টরির জন্য না। কোনো মাস্ক সেট করলে তা এরপর তৈরী করা নতুন ফাইলগুলোর উপর কার্যকর হয়। একটি উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ umask
0002
me@howtocode-pc:~$ > foo.txt
me@howtocode-pc:~$ ls -l foo.txt
-rw-rw-r-- 1 me me 0 Oct  7 10:49 foo.txt
```

প্রথমে আমরা কোনো আর্গুমেন্ট ছাড়া **umask** কমান্ডটি দিয়েছি। এতে আমরা কোনো মাস্ক তৈরী করিনি, বরং আগে থেকে থাকা মাস্কটি দেখলাম। এবং সেই মাস্কের মান ছিল 0002। এরপর আমরা `> foo.txt` কমান্ড দিয়ে foo.txt নামে একটা ফাইল তৈরী করলাম এবং `ls -l foo.txt` দিয়ে এর পারমিশন দেখে নিলাম। এবার আমরা প্রথম পুরনো foo.txt ফাইলটি মুছে দেবো। তারপর 0000 মাস্ক সেট করবো বা বলতে পারি মাস্কিং বন্ধ করবো এবং নতুন করে foo.txt ফাইলটি তৈরী করে এর পারমিশন দেখাবো:

```
me@howtocode-pc:~$ rm foo.txt
me@howtocode-pc:~$ umask 0000
me@howtocode-pc:~$ > foo.txt
me@howtocode-pc:~$ ls -l foo.txt
-rw-rw-rw- 1 me me 0 Oct  7 11:01 foo.txt
```

এবার আমরা দেখেছি প্রথমবার 0002 মাস্কের জন্য ওয়ার্ল্ডের জন্য রাইট পারমিশন বাদ গিয়েছিল। কিন্তু এবার মাস্কিং অফ করায় ডিফল্টভাবে সেটি আছে। আসুন দেখা যাক:

ধাপ	বিস্তারিত
মূল ফাইল মোড	--- rw- rw- rw-
মাস্ক	0 0 0 2
বাইনারি	000 000 000 010
ফলাফল	--- rw- rw- r--

আমরা ফাইলমোডের প্রথম অংশটা অর্থাৎ প্রথম তিন অংক এখন আমলে আনবো না। পরের লেসনে সেটি নিয়ে কথা হবে। তারপরেই আমরা দেখছি ওনার ইউজার ও গ্রুপের এবং ওয়ার্ল্ড সবার জন্য ডিফল্ট পারমিশন rw-। এরপর আমরা 0002 মাস্ক ব্যবহার করেছি। এটি অকটাল নাম্বার। আমরা যদি এর প্রতিটি অংককে তিন অংকের বাইনারিতে



রূপান্তর করি তাহলে 0 হবে 000 এবং 2 হবে 10। এভাবে আমরা পেয়েছি: 000 000 000 010। এবার হিসেবটা সোজা। ডিফল্ট ফাইলমোডে যার নীচে 1 থাকবে সেটি বাদ যাবে। এখানে ওয়ার্ল্ডের রাইট পারমিশনের নীচে এক ছিল ফলে এটি বাদ গিয়েছে। একইভাবে 0022 ব্যবহার করলে ওয়ার্ল্ডের সাথে গ্রুপেরও রাইট পারমিশন বাদ যাবে।

সাধারণত আপনার মাস্কিং পরিবর্তনের দরকার পরবে না। তবে কখনো কখনো উচ্চপর্যায়ের নিরাপত্তা নিশ্চিত করতে প্রয়োজন হতে পারে। আপনি আরো কয়েকটা মাস্কিং ড্যালু নিয়ে টেস্ট করতে পারেন। সবশেষে আপনার ডিফল্ট মাস্কিং ড্যালুতে(যেমন আমার ক্ষেত্রে 0002) ফিরে যেতে ভুলবেন না।

## বিশেষ পারমিশন

আমরা আগের [লেসনে](#) দেখেছি, মাস্কিং এর সময় চার অঙ্কের অকটাল সংখ্যা ব্যবহার করেছি। এর প্রথম অংকটি কিছু স্পেশাল পারমিশন নির্দেশ করতে কাজে লাগে। প্রথম অংক বা বিটকে ব্যবহার করে তিনটি স্পেশাল পারমিশন দেয়া হয়:

- **setuid:** এর অকটাল বিট 4000 এবং কোনো প্রোগ্রামকে এই পারমিশন দিতে আমরা লিখতে পারি:

```
chmod u+s program_name
```

কমান্ডটি দিলে দেখা যাবে ইউজারের এক্সিকিউশন পারমিশনের জায়গায় যে x ছিল সেটি বদলে s হয়ে গেছে।

এই পারমিশন দিলে সিস্টেম দেখবে প্রোগ্রামটার আসল ওনার বা মালিক ইউজার কে। এবং তার সব পারমিশন সুদূর প্রোগ্রামটি চালাতে দেবে। এর একটা বাস্তব ব্যবহার দেখলে প্রয়োজনীয়তা বুঝতে পারবেন। যেমন মনে করুন 'passwd' কমান্ডটির কথা। এই কমান্ডটি দিয়ে কেউ তার পাসওয়ার্ড পরিবর্তন করতে পারবেন। কিন্তু পাসওয়ার্ডগুলো কোথায় সংরক্ষিত থাকে? এটা থাকে **/etc/shadow** ফাইলে। এই ফাইলটার মালিক রুট। এখন ভাবুন আপনি আপনার পাসওয়ার্ড পরিবর্তন করতে চাচ্ছেন। কিন্তু কিভাবে করবেন যদি ওই ফাইলে আপনার অধিকার না থাকে? সিক্যুরিটির জন্য একজন সাধারণ ইউজারকে সেই ফাইলে অধিকার দেওয়া নিরাপদ না। তারচেয়ে আপনাকে যদি `passwd` কমান্ডটি এভাবে ব্যবহার করতে দেয়া হয় যে আপনি ওটা এমনভাবে ব্যবহার করতে পারবেন যে ওটা ব্যবহারের সময় আপনি রুটের মতই সবকিছু করতে পারবেন। তাহলে বরং সুবিধা হয়। আপনি তখন নিজের পাসওয়ার্ড চেঞ্জ ছাড়া আর কোনো কারনে **/etc/shadow** ব্যবহার করতে পারবেন। না। এজন্য প্রায় সব লিনাক্স সিস্টেমে `passwd` কমান্ডটিকে **setuid** পারমিশন দিয়ে রাখা হয়। ফলে যখন সাধারণ ইউজার ওই কমান্ডটি ব্যবহার করে সুপারইউজার বা রুটের মতই ব্যবহার করতে পারে, প্রয়োজনীয় ফাইলে এক্সেস নিতে পারে অন্যসময় পারে না।

বুঝতেই পারছেন এটি একটি সাংঘাতিক পারমিশন। খুব সামান্য কিছু প্রোগ্রাম ছাড়া মোটেই এই পারমিশন দেয়া হয় না।

- **setgid:** এর অকটাল বিট 2000 এবং এটি গ্রুপের উপর কার্যকর হয়। এবং এটি প্রোগ্রাম নয় বরং ডিরেক্টরির ক্ষেত্রে প্রযোজ্য। আপনি এই পারমিশন দিতে পারেন এভাবে:

```
chmod g+s directory
```

কমান্ডটি দিলে গ্রুপের এক্সিকিউশন পারমিশনের x পরিবর্তিত হয়ে s হয়ে যাবে।

আপনি যখন কোথাও কোনো ফাইল বা ডিরেক্টরি তৈরি করেন, সেটি আপনার নামের একটি গ্রুপের মালিকানায় থাকে। কিন্তু আপনি একটা ডিরেক্টরি তৈরি করলেন আর তার ওনার গ্রুপের নাম মনে করুন **shared**। এবার এটিকে আপনি **setgid** পারমিশন দিলেন। ফলে হবে কি, এর মধ্যে আপনি যখন নতুন কোনো ফাইল তৈরি করবেন এটি কিন্তু আর আপনার নামের গ্রুপে থাকবে না, বরং ওই ডিরেক্টরির মালিক অর্থাৎ **shared** গ্রুপের অধীনে থাকবে। সাধারণত একাধিক ইউজার নিজেদের মধ্যে ফাইল শেয়ারিং এর পারমিশন এর ঝামেলা এড়াতে এটি ব্যবহার করে।

- **sticky bit:** এই পারমিশনটিও ডিরেক্টরির উপর কার্যকর হয়। এই পারমিশন কোনো ডিরেক্টরিকে দিতে ব্যবহার

করুন:

```
chmod +t directory
```

এটি দিলে দেখবে ওয়ার্ল্ডের এক্সিকিউশন পারমিশনের x এর জায়গায় t এসেছে।

কোনো ডিরেক্টরিকে এই পারমিশন দিলে তার মধ্যের কোন ফাইল ডিলিট বা তার নাম পরিবর্তন করতে পারবে না সবাই। ডিলিট বা নাম পরিবর্তন করতে হয় তাকে এই ডিরেক্টরের মালিক হতে হবে, অথবা ওই ফাইলটির মালিক হতে হবে অথবা হতে হবে একজন সুপারইউজার। শেয়ারড ফোল্ডারে অনাকাঙ্ক্ষিত অনুপ্রবেশ ম্যানিপুলেশন ঠেকাতে এটি ব্যবহার করা হয়।

## ওনার ইউজার ও গ্রুপ পরিবর্তন

কোনো ফাইল বা ডিরেক্টরির মালিকানা পরিবর্তনের দুটো কমান্ড আছে। **chown** ও **chgrp**। অবশ্য **chown** একাই ওনার ইউজার ও গ্রুপ পরিবর্তনে সক্ষম। **chgrp** শুধু ওনার গ্রুপ পরিবর্তন করতে পারে। তবুও কখনো কখনো **chgrp** ব্যবহার করা হয় সরল কাঠামোর জন্য। প্রথমে দেখা যাক **chgrp** এর ব্যবহার:

```
me@howtocode-pc:~$ > foo.txt
me@howtocode-pc:~$ ls -l foo.txt
-rw-rw-r-- 1 me me 0 Oct  7 22:51 foo.txt
me@howtocode-pc:~$ chgrp video foo.txt
me@howtocode-pc:~$ ls -l foo.txt
-rw-rw-r-- 1 me video 0 Oct  7 22:51 foo.txt
```

প্রথমে আমরা foo.txt নামের একটি ফাইল তৈরি করেছি। তারপর `ls -l foo.txt` কমান্ড দিয়ে দেখলাম এটির মালিক ইউজার me এবং মালিক গ্রুপও me নামের একটি গ্রুপ। এখন আমরা video নামের একটি গ্রুপকে মালিকানা দিতে চাই। তাই `chgrp video foo.txt` কমান্ডটি দিয়েছি। অর্থাৎ chgrp এর কমান্ড কাঠামোটি হল:

```
chgrp group_name file_or_directory...
```

এরপর আমরা আবার `ls -l foo.txt` কমান্ড দিয়ে দেখলাম যে গ্রুপ পরিবর্তিত হয়ে video হয়ে গেছে।

**chown** কমান্ডটি ব্যবহার করতে সুপারইউজার বা রুট পারমিশন প্রয়োজন হয়। এর কমান্ড কাঠামোটি এরকম:

```
chown owner:group file_or_directory...
```

এই **owner:group** এর জায়গায় শুধু ওনার এর নাম বা ':' চিহ্নের পর শুধু গ্রুপের নাম বা মাঝখানে ':' চিহ্ন রেখে ইউজার ও গ্রুপ দুটোর নামই দেয়া যায়। আসুন এর আর্গুমেন্টের কয়েকটা উদাহরণ দেখা যাক:

আর্গুমেন্ট	রেজাল্ট
bob	ফাইলটির ওনার ইউজার হবে bob নামের ইউজার। ওনার গ্রুপ অপরিবর্তিত থাকবে।
bob:users	ফাইলটির ওনার ইউজার হবে bob এবং ওনার গ্রুপ হবে users নামের গ্রুপ।
:admins	ফাইলটির ওনার ইউজার অপরিবর্তিত থাকবে কিন্তু ওনার গ্রুপ হবে admins।
bob:	ফাইলটির ওনার ইউজার হবে bob এবং গ্রুপ হবে bob এর লগিন গ্রুপ, যেটি কিনা সাধারণত তার নামের গ্রুপটি হয়।

## পরিচয় পরিবর্তন

বিভিন্ন সময় আমাদের ভিন্ন ইউজার হিসেবে কাজ করতে হতে পারে। বিশেষ করে সুপারইউজার হিসেবে বিভিন্ন এডমিনিস্ট্রেটিভ কাজ আমাদের করতে হয়। তাছাড়া অন্য একজন ইউজার হওয়ারও প্রয়োজন হতেই পারে। আপনি যেটাই করতে চান আপনাকে সেই ইউজারের পাসওয়ার্ড জানতে হবে। পাসওয়ার্ড জানা থাকলে আপনি su ও sudo কমান্ডদ্বয়ের মাধ্যমে আইডেন্টিটি বা পরিচয় পরিবর্তন করতে পারেন।

### su

**su** কমান্ডটির মাধ্যমে আমরা অন্য ইউজারে পরিবর্তিত হতে পারি। এর সাধারণ কমান্ডকাঠামো এরকম:

```
su username
```

অর্থাৎ **su** কমান্ডের আর্গুমেন্ট হিসেবে যে ইউজারে পরিবর্তিত হতে চাই তার নাম দিতে হবে। এরপর টার্মিনাল আপনার কাছে ওই ইউজারের পাসওয়ার্ড চাইবে। একটা ব্যাপার লক্ষ্যণীয়, টার্মিনালে পাসওয়ার্ড লিখলেও কিছু দেখাবে না। কিন্তু পাসওয়ার্ড লিখে এন্টার চাপলে কাজ করবে। আমরা যদি কমান্ডটির অপশন হিসেবে -l ব্যবহার করি তাহলে ওই ইউজারের সম্পূর্ণ এনভায়রনমেন্ট সহ লোড হবে। আর যদি কোনো আর্গুমেন্ট ছাড়াই **su** কমান্ডটি দিই তবে সুপারইউজার হিসেবে লগিন করবে। এক্ষেত্রেও সুপারইউজারের পাসওয়ার্ড জানা প্রয়োজন। সুপারইউজারে আইডেন্টিটি পরিবর্তন করলে প্রম্পট এর শেষের '\$' চিহ্নটি '#' চিহ্নে পরিবর্তিত হবে। কোনো ইউজারে পরিবর্তিত হওয়ার পর কাজ শেষে **exit** কমান্ড দিয়ে পূর্বের ইউজারে ফিরে আসতে পারি।

### sudo

**sudo** কমান্ডের মাধ্যমে সুপারইউজার না হয়েই সুপারইউজারের মত কাজ করা যায়। এডমিনিস্ট্রেটর **sudo** কমান্ডকে এমনভাবে কনফিগার করতে পারেন যে সে সকল সুপারইউজার কমান্ড বা নির্দিষ্ট কিছু সুপারইউজার কমান্ড এক্সিকিউট করতে পারে। এজন্য তাকে সুপারইউজারের পাসওয়ার্ড জানতে হবে না। এর কমান্ড কাঠামোটি এরকম:

```
sudo command
```

অর্থাৎ, যে কমান্ডটি সুপারইউজার হিসেবে কার্যকর করতে চাই তার সামনে **sudo** বসাতে হবে। এরপর তাকে তার নিজের পাসওয়ার্ড দিতে বলা হবে।

যারা উবুন্টু বা উবুন্টুবেজড ডিস্ট্রিবিউশন ব্যবহার করেন তারা সাধারণভাবে সুপারইউজার হিসেবে লগিন করতে পারেন না। কারণ ইন্সটলেশনের সময় সুপারইউজারের কোনো পাসওয়ার্ড সেট করতে দেয়া হয় না। তার বদলে প্রথম ইউজার ডিফল্টভাবে sudo ব্যবহারের পারমিশন পায়। সে sudo এর মাধ্যমে সকল সুপারইউজার প্রিভিলেজ পায়। এর মাধ্যমে নিরবিচ্ছিন্ন সুপারইউজার একাউন্ট ব্যবহারের সিক্যুরিটি রিস্ক অনেকটাই এড়ানো যায়।



## পাসওয়ার্ড পরিবর্তন

**passwd** কমান্ড দিয়ে আপনি আপনার পাসওয়ার্ড পরিবর্তন করতে পারেন। সুপারইউজার হলে পরিবর্তন করতে পারেন অন্য ইউজারের পাসওয়ার্ডও। কোনো ইউজারের পাসওয়ার্ড পরিবর্তন করার জন্য প্রয়োজনীয় কমান্ডকাঠামোটি এরকম:

```
passwd user
```

তবে নিজের পাসওয়ার্ড পরিবর্তন করতে কোনো আর্গুমেন্ট ছাড়াই **passwd** কমান্ডটি দিলেই হয়। তখন প্রথমে আপনার পুরাতন পাসওয়ার্ড দিতে বলবে। সঠিকভাবে পুরাতন পাসওয়ার্ড দিলে তবেই নতুন পাসওয়ার্ড দিতে বলবে। খুব ছোট পাসওয়ার্ড, আগের পাসওয়ার্ডই নতুন পাসওয়ার্ড হিসেবে দিতে চাইলে বা পাসওয়ার্ডটি ডিকশনারীর কোনো শব্দ হলে সেটি নেবে না।

## অধ্যায় - সাত

### প্রসেস

আধুনিক অপারেটিং সিস্টেমগুলোকে আমরা দেখি একসাথে একাধিক কাজ করতে পারে। একে বলে মাল্টিটাস্কিং সিস্টেম। আসলে এই অপারেটিং সিস্টেমগুলো অতিদ্রুত একটা কাজ একটু করে অন্য একটা আরেকটু করে আবার আরেকটা খানিকটা করে বলে এদের মাল্টিটাস্কিং বলে মনে হয়। এই যে অতি দ্রুত একটা কাজ থেকে আরেকটা কাজ এ যাওয়া, কোনটা কখন করা হবে সেটা কার্নেল ঠিক করে। এবং কার্নেল এটা ঠিক করে প্রসেসের মাধ্যমে। সে একাধিক প্রসেস এর থেকে বেছে নেয় কোনটা কখন করবে। এই অধ্যায়ে আমরা প্রসেস নিয়ে কথা বলবো।

- **প্রসেস এর প্রাথমিক ধারণা:** প্রসেস কী এবং কীভাবে কাজ করে।
- **প্রসেস দেখা:** **ps** ও **top** কমান্ডের ব্যবহার।
- **প্রসেস নিয়ন্ত্রণ:** প্রসেস নিয়ন্ত্রণ সম্পর্কিত ধারণা।
- **সিগন্যাল:** সিগন্যাল সম্পর্কিত ধারণা এবং **kill** ও **killall** কমান্ডের ব্যবহার।



## প্রসেস এর প্রাথমিক ধারণা

### প্রসেস কি?

সাদামাটা কথায় যা কম্পিউটারের কাছে কাজ, তাই প্রসেস। তা চালু থাকা কোনো প্রোগ্রাম হতে পারে। বা যেকোনো ধরনের কাজ।

### কীভাবে প্রসেস কাজ করে?

যখন সিস্টেম চালু হয়, কার্নেল নিজের কিছু কাজ শুরু করে প্রসেস হিসেবে। তারপর একটা প্রোগ্রাম চালু করে যার নাম **init**। এরপর **init** প্রোগ্রামটি **/etc** ডিরেক্টরিতে থাকা কিছু শেলস্ক্রিপ্ট চালু করে যাদের ইনিটস্ক্রিপ্ট(**init script**) বলা হয়। এই স্ক্রিপ্টগুলো কিছু সার্ভিস প্রোগ্রাম চালু করে যেগুলো নেটওয়ার্কিংসহ বিভিন্ন বিষয় পরিচালনা করে। এগুলো অধিকাংশই ডেমন(**daemon**) প্রোগ্রাম অর্থাৎ যাদের ইউজার ইন্টারফেস নেই বরং ব্যাকগ্রাউন্ডে কাজ করে। মানে আমরা যদি লগিন নাও করি কম্পিউটার কিছু কাজকর্মে ব্যস্ত থাকে। এরপর আমরা কোনো প্রোগ্রাম চালালে সেগুলো প্রসেসে যুক্ত হয়। এবং সেই প্রোগ্রামগুলো আরো প্রোগ্রাম তথা প্রসেস চালু করতে পারে। তখন বলা হয় প্যারেন্ট প্রসেস চাইল্ড প্রসেস তৈরী করছে।

কার্নেল প্রত্যেক প্রসেস সম্পর্কে তথ্য রাখে যেন তাদের ঠিকমত ব্যবহার করতে পারে। যেমন প্রত্যেক প্রসেসকে সে একটি নম্বর দিয়ে দেয় যাকে PID বলে। **init** সবার আগে চালু হয় বলে এর PID হয় 1। তাছাড়াও কোন প্রসেসের কতটুকু মেমরী লাগবে, বা তা চালু হতে প্রস্তুত কিনা এগুলোর খবরও কার্নেলকে রাখতে হয়।

একটা প্রসেস আরো প্রসেস তৈরী করতে পারে। নতুন জন্ম দেয়া প্রসেসটিকে চাইল্ড প্রসেস বলা হয় আর তার জন্মদাতা প্রসেসকে প্যারেন্ট প্রসেস।

## প্রসেস দেখা

প্রসেস দেখার সবচেয়ে সাদামাটা কমান্ডটি হল **ps**। এর সাথে ব্যবহারযোগ্য অনেক অপশন আছে (man ps দিয়ে দেখে নিতে পারেন) তবে শুধু ps দিয়েও আপনি দেখতে পারেন:

```
me@howtocode-pc:~$ ps
  PID TTY          TIME CMD
 19394 pts/5    00:00:00 bash
 19575 pts/5    00:00:00 ps
```

ফলাফল হিসেবে আমরা দুটো প্রসেস পেয়েছি। প্রথম প্রসেসটার প্রসেস আইডি বা **PID** হল 19394। TTY হল 'teletype' এর সংক্ষিপ্ত রূপ। এই প্রসেসদুটি টেলিটাইপ কন্ট্রোলিং টার্মিনাল 5 এ কাজ করছে। TTY আমাদের কাছে গুরুত্বপূর্ণ কিছু না। তারপরেই TIME। টাইম বলতে এখানে ওই প্রসেসটি প্রসেসরের কতটা সময় ব্যয় করে তাই নির্দেশ করে। আর শেষে CMD এর ঘরে থাকে কমান্ডটি।

এ বড় সাদামাটা লাগছে। তাইতো? আসুন কিছু অপশন সহ চেষ্টা করি। প্রথমে আমরা 'x' অপশনটি ব্যবহার করবো। অন্য কমান্ডের মত ps কমান্ডের অপশনগুলোর সামনে '-' বা '--' থাকে না।

```
me@howtocode-pc:~$ps x
  PID TTY          STAT TIME COMMAND
 2191 ?           S1    0:00 /usr/bin/gnome-keyring-daemon --daemonize --login
 2193 ?           Ss    0:00 init --user
 2269 ?           Ss    0:00 ssh-agent -s
 2291 ?           Ss    0:00 dbus-daemon --fork --session --address=unix:abstract=/tmp/dbus
 2298 ?           Ss    0:00 upstart-event-bridge
 2311 ?           S     0:00 upstart-file-bridge --daemon --user
 2313 ?           S     0:00 upstart-dbus-bridge --daemon --system --user --bus-name system
 2315 ?           S     0:00 upstart-dbus-bridge --daemon --session --user --bus-name sessi
 2317 ?           Ss1   0:06 /usr/bin/ibus-daemon --daemonize --xim
 2331 ?           Ss1   0:01 /usr/lib/gnome-settings-daemon/gnome-settings-daemon
 2336 ?           S1    0:00 /usr/lib/gvfs/gvfsd
 2341 ?           Ss1   0:00 /usr/lib/at-spi2-core/at-spi-bus-launcher --launch-immediately
 2342 ?           Ss1   0:00 gnome-session --session=gnome
 2352 ?           S1    0:00 /usr/lib/ibus/ibus-dconf
 2356 ?           S1    0:00 /usr/lib/gvfs/gvfsd-fuse /run/user/1001/gvfs -f -o big_writes
 2358 ?           S1    0:00 /usr/lib/ibus/ibus-ui-gtk3
 2361 ?           S     0:00 /bin/dbus-daemon --config-file=/etc/at-spi2/accessibility.conf
 2368 ?           S1    0:02 /usr/lib/ibus/ibus-x11 --kill-daemon
 2372 ?           S1    0:00 /usr/lib/at-spi2-core/at-spi2-registryd --use-gnome-session
 2428 ?           S<1   0:00 /usr/bin/pulseaudio --start --log-target=syslog
 2431 ?           S1    0:00 /usr/lib/dconf/dconf-service
 2434 ?           S1    0:01 /usr/lib/ibus/ibus-engine-simple
 2444 ?           S     0:00 syndaemon -i 1.0 -t -K -R
 2450 ?           S1    0:00 /usr/lib/gnome-settings-daemon/gsd-printer
 2455 ?           S1    1:42 /usr/bin/gnome-shell
 2882 ?           S1    0:00 /usr/lib/gnome-shell/gnome-shell-calendar-server
```

2888 ?	Sl	0:00 /usr/lib/evolution/evolution-source-registry
2892 ?	Sl	0:00 /usr/lib/gvfs/gvfsd-metadata
2897 ?	SLl	0:02 /usr/lib/gnome-online-accounts/goa-daemon
2902 ?	Sl	0:00 /usr/lib/telepathy/mission-control-5
2907 ?	Sl	0:00 /usr/lib/evolution/evolution-calendar-factory
2912 ?	Sl	0:00 /usr/lib/gvfs/gvfs-udisks2-volume-monitor
2928 ?	Sl	0:00 /usr/lib/gvfs/gvfs-goa-volume-monitor
2933 ?	Sl	0:00 /usr/lib/gvfs/gvfs-mtp-volume-monitor
2939 ?	Sl	0:00 /usr/lib/gvfs/gvfs-afc-volume-monitor
2944 ?	Sl	0:00 /usr/lib/gvfs/gvfs-gphoto2-volume-monitor
2949 ?	Sl	0:02 /usr/bin/nautilus --no-default-window
2957 ?	Sl	0:16 /usr/lib/tracker/tracker-store
2958 ?	SNl	0:34 /usr/lib/tracker/tracker-miner-fs
2962 ?	Sl	0:00 tilda
2965 ?	Sl	0:08 cairo-dock
2986 ?	Sl	0:10 yakuake
2996 ?	Sl	0:00 /usr/lib/telepathy/telepathy-logger
2998 ?	S	0:00 /usr/lib/x86_64-linux-gnu/gconf/gconfd-2
3011 ?	S	0:00 gnome-pty-helper
3012 pts/10	Ss+	0:00 /bin/bash
3068 ?	S	0:00 python /usr/lib/x86_64-linux-gnu/cairo-dock/cairo-dock-launcher
3100 ?	Sl	0:00 /usr/lib/gvfs/gvfsd-burn --spawner :1.4 /org/gtk/gvfs/exec_spa
3133 ?	Sl	0:00 /usr/lib/gvfs/gvfsd-trash --spawner :1.4 /org/gtk/gvfs/exec_sp
3227 ?	Sl	0:00 /usr/lib/gvfs/gvfsd-computer --spawner :1.4 /org/gtk/gvfs/exec
3238 ?	Ss	0:00 kdeinit4: kdeinit4 Running...
3241 ?	S	0:00 kdeinit4: klauncher [kdeinit] --fd=8
3244 ?	Sl	0:00 kdeinit4: kded4 [kdeinit]
3580 ?	Sl	0:01 zeitgeist-datahub
3585 ?	Sl	0:00 /usr/bin/zeitgeist-daemon
3646 ?	Sl	0:00 /usr/lib/x86_64-linux-gnu/zeitgeist-fts
3653 ?	S	0:00 /bin/cat
3657 ?	Sl	0:00 /usr/lib/gvfs/gvfsd-http --spawner :1.4 /org/gtk/gvfs/exec_spa
3667 ?	Sl	0:00 /usr/bin/kglobalaccel
3685 pts/12	Ss+	0:00 /usr/bin/tmux
3690 ?	Ss	0:01 /usr/bin/tmux
3691 pts/13	Ss	0:00 -bash
3698 ?	Sl	0:00 /usr/bin/knotify4
4057 ?	Sl	0:00 update-notifier
5200 ?	Sl	0:00 /usr/lib/x86_64-linux-gnu/deja-dup/deja-dup-monitor
10038 ?	Sl	0:00 /usr/bin/kactivitymanagerd
16237 ?	Sl	0:04 okular /home/me/sherlock.pdf --icon okular -caption Okular
18203 pts/13	S+	0:05 emacs -nw 1.7.2.viewprocess.md
18808 pts/15	Ss	0:00 -bash
18837 pts/15	Sl+	0:01 cmus
19394 pts/5	Ss	0:00 /bin/bash --noediting -i
21797 pts/5	R+	0:00 ps x

আমরা লম্বা একটা লিস্ট পেয়েছি প্রসেসের। তারসাথে STAT নামের নতুন একটা ঘর যোগ হয়েছে। 'x' অপশন দিয়ে আমরা বলেছি যেকোনো কন্টোলিং টার্মিনাল থেকে চালু প্রোগ্রাম দেখাতে। STAT যেটি কিনা 'state' এর সংক্ষিপ্তরূপ প্রসেসটির বর্তমান অবস্থা সম্পর্কে গুরুত্বপূর্ণ তথ্য দেয় সংক্ষিপ্ত ভাবে। আসুন এই অক্ষরগুলোর অর্থ জেনে নেয়া যাক:

নির্দেশক	অর্থ
R	Running Process। অর্থাৎ যে প্রসেস এখনো কাজ করছে প্রসেসরে বা এখন কাজ করতে প্রস্তুত।
S	Sleeping Process। যে প্রসেসটা এখন কাজ না করে অপেক্ষা করছে কিছু একটা ঘটার। সেটা যেকোনো ধরনের সংকেত হতে পারে। কোনো বাটন চাপা বা কোনো নেটওয়ার্ক প্যাকেট পাওয়া।
D	Uninterruptible Sleep। এমন স্লিপিং প্রসেস যা কোনোরকম ইনপুট বা আউটপুটের জন্য অপেক্ষা করছে এবং যাকে বন্ধ করা যায় না। যেমন ডিস্ক ড্রাইভ।
T	Stopped Process। যে প্রসেসকে বন্ধ করতে বলা হয়েছে। আমরা পরে এ নিয়ে বিস্তারিত জানবো।
Z	Zombie Process। এগুলো এমনসব চাইল্ড প্রসেস যা বন্ধ করা হয়েছে কিন্তু তার প্যারেন্ট প্রসেস এখনো তাকে মুছে ফেলেনি।
<	অতিগুরুত্বপূর্ণ প্রসেস। এমন প্রসেস যাকে প্রসেসরে বেশি সময় নিতে সুযোগ দেয়া হয়।(প্রসেসের ক্ষেত্রে niceness বলে একটি টার্ম আছে। এধরনের প্রসেসকে less nice বলা হয় কারন এরা নিজেরা বেশি সময় ব্যবহার করে এবং অন্যদের সময় কম দেয়।)
N	একটি কম গুরুত্বপূর্ণ বা nice process। গুরুত্বপূর্ণ প্রসেসগুলোর পর এরা প্রসেসরে সময় পায়।

আরো তথ্যসমৃদ্ধ প্রসেস লিস্ট আমরা দেখতে পারি এভাবে:

```
me@howtocode-pc:~$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.0	33832	3160	?	Ss	11:29	0:02	/sbin/init
root	2	0.0	0.0	0	0	?	S	11:29	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	11:29	0:00	[ksoftirqd/0]
root	5	0.0	0.0	0	0	?	S<	11:29	0:00	[kworker/0:0H]
root	7	0.0	0.0	0	0	?	S	11:29	0:01	[rcu_sched]
root	8	0.0	0.0	0	0	?	S	11:29	0:01	[rcuos/0]
root	9	0.0	0.0	0	0	?	S	11:29	0:00	[rcuos/1]
root	10	0.0	0.0	0	0	?	S	11:29	0:00	[rcuos/2]
root	11	0.0	0.0	0	0	?	S	11:29	0:00	[rcuos/3]
root	12	0.0	0.0	0	0	?	S	11:29	0:00	[rcu_bh]
root	13	0.0	0.0	0	0	?	S	11:29	0:00	[rcuob/0]
root	14	0.0	0.0	0	0	?	S	11:29	0:00	[rcuob/1]
....										

আমরা পুরো লিস্টটি এখানে দিইনি। এখানে আমরা USER নামে ঘরে দেখতে পারছি কোন ইউজার এই প্রসেসটি চালু করেছে। %CPU এবং %MEM নির্দেশ করছে শতকরা কতটুকু প্রসেসর ও মেমরি প্রসেসটি খরচ করেছে। VSZ ভার্চুয়াল মেমরির পরিমাণ দেখাচ্ছে। RSS দেখাচ্ছে Resident Set Size অর্থাৎ RAM এ কতটুকু জায়গা প্রসেসটি নিচ্ছে(এটি কিলোবাইটে প্রকাশ করা হচ্ছে)। START দিয়ে বোঝানো হচ্ছে কখন প্রসেসটি চালু করা হয়েছে।

## ডায়নামিক্যালি প্রসেস দেখা

**ps** কমান্ডটি আমরা যখন ব্যবহার করি ঠিক সেই মুহূর্তের প্রসেসের একটি চিত্র আমরা দেখতে পাই। আমরা যদি চাই এটা ডায়নামিক্যালি দেখতে তাহলে **top** কমান্ডটি ব্যবহার করতে হবে:

```
me@howtocode-pc:~$ top
```

```
top - 12:43:00 up 1:13, 3 users, load average: 0.07, 0.19, 0.27
Tasks: 178 total, 2 running, 176 sleeping, 0 stopped, 0 zombie
%Cpu(s): 12.7 us, 3.0 sy, 4.6 ni, 75.2 id, 4.6 wa, 0.0 hi, 0.1 si, 0.0 st
KiB Mem: 3915312 total, 2698912 used, 1216400 free, 283676 buffers
KiB Swap: 2097148 total, 0 used, 2097148 free. 1458552 cached Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
8	root	20	0	0	0	0	R	6.1	0.0	0:01.50	rcuos/0
2455	me	20	0	1477500	168964	33464	S	6.1	4.3	2:32.69	gnome-shell
29407	me	20	0	29404	1516	1064	R	6.1	0.0	0:00.01	top
1	root	20	0	33832	3160	1472	S	0.0	0.1	0:02.02	init
2	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kthreadd
3	root	20	0	0	0	0	S	0.0	0.0	0:00.02	ksoftirqd/0
5	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/0:0H
7	root	20	0	0	0	0	S	0.0	0.0	0:01.64	rcu_sched
9	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuos/1
10	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuos/2
11	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuos/3
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcu_bh
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/0
14	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/1
15	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/2
16	root	20	0	0	0	0	S	0.0	0.0	0:00.00	rcuob/3
17	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
18	root	rt	0	0	0	0	S	0.0	0.0	0:00.02	watchdog/0
19	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	khelper
20	root	20	0	0	0	0	S	0.0	0.0	0:00.00	kdevtmpfs
21	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	netns
22	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	writeback
23	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kintegrityd
24	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	bioaset
25	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kworker/u9:0
26	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	kblockd
27	root	0	-20	0	0	0	S	0.0	0.0	0:00.00	ata_sff
28	root	20	0	0	0	0	S	0.0	0.0	0:00.00	khudb

কমান্ডটি দিলে টপ প্রোগ্রামটি চালু হবে। এর প্রথম অংশে সিস্টেম সম্পর্কিত বেশ কিছু গুরুত্বপূর্ণ তথ্য দেবে। আর দ্বিতীয় অংশে প্রসেসের লিস্ট। এটি প্রতি তিনসেকেন্ডে একবার আপডেট হয়। এবার আমরা কমান্ডটির বিভিন্ন অংশের অর্থ জেনে নিই:

সারি	অংশ	অর্থ
1	top	প্রোগ্রামের নাম ।
1	12:43:00	বর্তমান সময় ।
1	up 1:13	কম্পিউটার কতক্ষণ চালু আছে বা আপটাইম ।
1	3 users	কতজন ইউজার লগড-ইন আছে ।
1	load average:	কতগুলো প্রসেস কাজ করার জন্য প্রস্তুত । এখানে তিনটি সংখ্যা আছে প্রথমটি দিয়ে গত ৬০সেকেন্ডের লোড এভারেজ এবং পরের দুটি দিয়ে যথাক্রমে ৫মিনিট ও ১৫মিনিটের লোড এভারেজ জানায় ।
2	Tasks:	এখানে বিভিন্নধরনের প্রসেসের মোট সংখ্যা দেখায় ।
3	Cpu(s):	এই সারিতে প্রসেসর সম্পর্কিত বিভিন্ন গুরুত্বপূর্ণ তথ্য দেখায় ।
3	12.7 us	শতকরা কতটুকু প্রসেসর কার্নেলছাড়া অন্যসব ইউজার দ্বারা ব্যবহৃত হচ্ছে ।
3	3.0 sy	শতকরা কতটুকু প্রসেসর কার্নেল বা সিস্টেম ব্যবহার করছে ।
3	4.6 ni	নাইস প্রসেস দ্বারা ব্যবহৃত প্রসেসরের শতকরা অংশ ।
3	75.2 id	শতকরা কতটুকু প্রসেসর অলস পড়ে আছে বা ব্যবহৃত হচ্ছে না ।
3	4.6 wa	শতকরা কতটুকু ইনপুট বা আউটপুটের অপেক্ষায় আছে ।
3	0.0 hi	হার্ডওয়্যার ইন্টারাপশনজনিত ব্যবহৃত প্রসেসর ।
3	0.1 si	সফটওয়্যার ইন্টারাপশনজনিত ব্যবহৃত প্রসেসর ।
3	0.0 st	হাইপারভাইজরে ব্যবহৃত সময় ।
4	Mem:	ব্যবহৃত মেমরি ।
5	Swap:	ব্যবহৃত সোয়াপ বা ভার্চুয়াল মেমরি ।

**top** এর বেশ কয়েকটি কীবোর্ড কমান্ড আছে । এটি বন্ধ করতে 'q' চাপুন ।

## প্রসেস নিয়ন্ত্রণ

আগের লেসনে আমরা প্রসেস দেখা বা মনিটর করার দুটি কমান্ড দেখেছি। এই লেসনে আমরা প্রসেস নিয়ন্ত্রণ করতে শিখবো। যেহেতু যেকোনো প্রসেস নিয়ে পরীক্ষা-নিরীক্ষা করা বিপদজনক হতে পারে তাই আমরা **xlogo** নামের একটি কমান্ডের উপর পরীক্ষা চালাবো। এই কমান্ডটি দিলে আপনি স্ক্রীনে X Window System যা কিনা আপনার গ্রাফিকাল ডেস্কটপের প্রাণ তার একটি লোগো দেখাবে। অতি নিরীহ এই প্রোগ্রামটি নিয়ে গবেষণা করতে বিপদের ঝুঁকি নেই। আপনার লিনাক্স সিস্টেমে গ্রাফিকাল এনভায়রনমেন্ট থাকে এই প্রোগ্রামটি ইন্সটল থাকারই কথা। না থাকলে gedit বা kwrite ব্যবহার করতে পারেন। এগুলো যথাক্রমে জিনোম ও কেডিই এর টেক্সট এডিটর।

আপনি কমান্ডটি দিলে প্রোগ্রামটি আলাদা একটি উইন্ডোতে চালু হবে এবং যতক্ষণ চলবে আপনার টার্মিনাল প্রম্পট ফিরে আসবে না। আপনি প্রোগ্রামটি বন্ধ করে দিলে প্রম্পট আসবে।

### প্রসেস ব্যহত(বন্ধ) বা ইন্টারাপ্ট করা

আগের মতই কমান্ড দিয়ে প্রোগ্রামটি চালু করুন। স্বাভাবিকভাবেই যতক্ষণ প্রোগ্রামটি চালু থাকবে প্রম্পট থাকবে না। কিন্তু আপনি এখন **Ctrl-c** চাপলে প্রোগ্রামটি বন্ধ হবে এবং প্রম্পট ফিরে আসবে। এভাবে আমরা অধিকাংশ প্রোগ্রামের প্রসেস ব্যহত, সোজা কথায় বন্ধ করতে পারি।

### প্রসেস ব্যাকগ্রাউন্ডে চালু করা

এমন প্রয়োজন হতে পারে(বা এমনটাই অধিকাংশ সময় প্রয়োজন হয়) যে আপনি একটি প্রোগ্রাম টার্মিনাল থেকে চালু করলেন এবং সেটি চালু রেখেই প্রম্পট ফিরে পেতে চান। সেক্ষেত্রে আপনার প্রোগ্রামটি ব্যাকগ্রাউন্ড প্রসেস হিসেবে চালু করতে হবে। এজন্য কমান্ডের শেষে আপনাকে **'&'** চিহ্ন ব্যবহার করতে হবে। অর্থাৎ আপনি যদি xlogo কমান্ডটি ব্যাকগ্রাউন্ডে চালু করতে চান তবে আপনাকে লিখতে হবে `xlogo &`। এই কমান্ডটি দিলে প্রোগ্রামটি চালু রেখেই আপনাকে প্রম্পট ফেরত দেবে:

```
me@howtocode-pc:~$ xlogo &
[1] 16725
```

কমান্ডটি দেয়ার পর দ্বিতীয় লাইনে দুটি সংখ্যা দেখিয়ে আমাদের প্রম্পট ফেরত দেবে টার্মিনাল। এর মধ্যে প্রথমে ব্রাকেটের মধ্যে সংখ্যাটি জব নম্বর এবং পরেরটি প্রসেস আইডি(PID)।

আমরা এখন **ps** কমান্ড দিলে বর্তমান প্রসেসগুলোর মধ্যে xlogo এর প্রসেসটি পাবো:

```
me@howtocode-pc:~$ ps
  PID TTY          TIME CMD
 16713 pts/14    00:00:00 bash
 16725 pts/14    00:00:00 xlogo
 16861 pts/14    00:00:00 ps
```

জব নাম্বার শেলের একটি বৈশিষ্ট্য। এটি ওই শেল দিয়ে চালু করা সকল প্রোগ্রাম বা জবের বা প্রসেস যাই বলুন তার একটি করে নাম্বার দিয়ে লিস্ট করে রাখে। **jobs** কমান্ডটি দিয়ে আমরা জবগুলো দেখতে পারি তাদের নাম্বার সহ:

```
me@howtocode-pc:~$ jobs
[1]+  Running                  xlogo &
```

আমরা আমাদের কমান্ডটি এক নাম্বার জব হিসেবে রানিং দেখছি।

## প্রসেস ফোরগ্রাউন্ডে আনা

একটা প্রসেস ব্যাকগ্রাউন্ডে যাওয়ার পর তার উপর কোনো নিয়ন্ত্রণ থাকে না। তাই প্রয়োজন হতে পারে তাকে আবার সামনে বা ফোরগ্রাউন্ডে আনার। এর জন্য আমাদের প্রথমে **jobs** কমান্ডটি দিয়ে প্রসেসটির জব নম্বর জানতে হবে:

```
me@howtocode-pc:~$ jobs
[1]+  Running                  xlogo &
```

এবার আমাদের **fg** কমান্ড ব্যবহার করতে হবে। এবং এর আর্গুমেন্ট হিসেবে % চিহ্নের সাথে জব নাম্বার দিতে হবে। আমাদের প্রোগ্রামের জব নম্বর 1, তাই আমরা লিখবো:

```
me@howtocode-pc:~$ fg %1
xlogo
```

পরের লাইনে প্রোগ্রামটার নাম আসবে এবং অন্যসব ফোরগ্রাউন্ড প্রসেসের মতই বন্ধ না করা পর্যন্ত প্রম্পট ফেরত আসবে না।

## প্রসেস বিরত(Pause) রাখা

কখনো কখনো দরকার হতে পারে একটি প্রসেস বন্ধ না করে শুধু পজ বা ফ্রীজ করে রাখার। কখনো কখনো একটি ফোরগ্রাউন্ড প্রসেসকে ব্যাকগ্রাউন্ডে পাঠানোর জন্য এটি ব্যবহৃত হয়। আমরা প্রথমে টার্মিনালে `xlogo` লিখে প্রোগ্রামটি চালু করি, তারপর Ctrl-z চেপে প্রসেসটাকে স্টপ বা পজ করি:

```
me@howtocode-pc:~$ xlogo
^Z
[1]+  Stopped                  xlogo
me@howtocode-pc:~$
```

দেখা গেলো প্রোগ্রামটিকে একটি স্টপড জব হিসেবে রেখে আমাদের কাছে প্রম্পট দেয়া হয়েছে। স্টপড মানে কিন্তু ব্যাকগ্রাউন্ড প্রসেস না। আমরা যদি এখন `xlogo` এর উইন্ডোটা টেনে ছোটবড় করি দেখবেন লোগোটিও ছোটবড় হওয়ার কথা কিন্তু হচ্ছে না। আমরা **bg** কমান্ড দিয়ে প্রসেসটিকে ব্যাকগ্রাউন্ডে আবার চালু করে দিলে সেটি কাজ করবে:



```
me@howtocode-pc:~$ bg %1  
[1]+  xlogo &
```

## সিগন্যাল

একটি নির্দিষ্ট প্রসেসকে বিশেষ কিছু করতে বলার পদ্ধতি হচ্ছে সিগন্যাল। তা হতে পারে আমরা তাকে বন্ধ হতে বা পজ করতে বলছি। আবার চালু হতে বলছি আরেকটি সিগন্যাল দিয়ে। প্রসেসকে সিগন্যাল পাঠাতে আমরা যে কমান্ডটি ব্যবহার করি সেটি হচ্ছে **kill**। এই কমান্ডের কমান্ডকাঠামো এরকম:

```
kill [-signal] PID_or_Job-ID...
```

অর্থাৎ প্রথমে কমান্ডটি তারপর যে সিগন্যালটি পাঠাতে চাই সেটি(নাম্বার বা নাম দিয়ে প্রকাশ করা) '-' চিহ্নসহ তারপর যে যে প্রসেসকে সিগন্যালটি দিতে চাই তাদের PID বা Job ID। কোনো সিগন্যালের নাম দেয়া না হলে ডিফল্টভাবে **TERM** সিগন্যালটি পাঠানো হয়।

আসুন, সচরাচর ব্যবহৃত সিগন্যালগুলো দেখি:

নাম্বার	নাম	অর্থ
1	HUP	হ্যাংআপ(Hangup)। এটি বলা যেতে পারে একটি ঐতিহ্যবাহী সিগন্যাল। পার্সনাল কম্পিউটার যখনও হাতে হাতে এসে পৌঁছায়নি তখন দূরবর্তী কম্পিউটারের সাথে ফোনলাইন বা মডেমের মাধ্যমে বহুমানুষ টার্মিনাল দিয়ে যুক্ত থাকত। এই টার্মিনালগুলো একধরনের কন্ট্রোলিং ডিভাইস। এই সিগন্যাল পাঠালে বুঝে নেয়া হত কন্ট্রোলিং টার্মিনালটি হ্যাংআপ করা হয়েছে। যেমনটা আমরা ফোনে করে থাকি। ফলাফল হিসেবে টার্মিনালে চালু থাকা সেশনগুলো বন্ধ করা হত। অনেক Daemon(যেমন Apache web server) এই সিগন্যালটিকে রিইনিশিয়ালাইজেশনের জন্য ব্যবহার করে। এই ডেমনগুলোকে এই সিগন্যাল পাঠালে তারা রিস্টার্ট নেয় এবং কনফিগারেশন ফাইলগুলো নতুন করে পড়ে।
2	INT	ইন্টারাপ্ট(Interrupt)। আমরা আগের লেসন Ctrl-c চেপে যা করেছি তাই করে। এই সিগন্যালটি পাঠালে অধিকাংশক্ষেত্রে প্রোগ্রামটি বন্ধ হয়ে যায়।
3	QUIT	কুইট(Quit)। নামেই এর অর্থ পরিষ্কার। প্রোগ্রাম কুইট করে। এটি একটি সিস্টেম সিগন্যাল অর্থাৎ সিস্টেম এটিকে ব্যবহার করে।
9	KILL	কিল(kill) একটি বিশেষধরনের সিগন্যাল। এটি প্রোগ্রামে পাঠানো হয় না। বরং যে প্রোগ্রামের জন্য এই সিগন্যাল দেয়া হয় কার্নেল তার প্রসেসটিকে সাথে সাথে বন্ধ করে দেয়। এক্ষেত্রে প্রোগ্রামটি কিছু সেভ করার বা অতিরিক্ত ডাটা পরিষ্কার করার সুযোগ পায় না।
11	SEGV	সেগমেন্টেশন ভায়োলেশন(Segmentation Violation)। এই সিস্টেম সিগন্যালটি তখন ব্যবহৃত হয় যখন কোনো প্রসেস ভুলভাবে মেমরী ব্যবহার করে বা এমন কোথাও তথ্য লিখতে যায় যেখানে তার অধিকার নেই।
15	TERM	টার্মিনেট(Terminate)। kill কমান্ডের সাথে কোনো সিগন্যাল না দেয়া হলে এই ডিফল্টভাবে এই সিগন্যালটি দেয়া হয়। এটি দিয়ে প্রসেসটি টার্মিনেট বা ধ্বংস করা হয়।
18	CONT	কন্টিনিউ(Continue)। STOP সিগন্যাল দিয়ে স্টপ করা বা পজ করা প্রসেসকে আবার সচল করে।
19	STOP	KILL সিগন্যালের মতই এই সিগন্যালটিও কার্নেল কার্যকর করে। তবে এক্ষেত্রে প্রসেসটিকে পজ করে দেয়া হয়।
20	TSTP	টার্মিনাল স্টপ(Terminal stop)। যখন টার্মিনালে Ctrl-z চাপা হয় তখন এই সিগন্যালটি টার্মিনাল পাঠায়। সাধারণ স্টপ সিগন্যালের মত এটি কার্নেল না বরং স্বয়ং প্রসেসটি কার্যকর করে। এবং চাইলে প্রসেসটি এটিকে অগ্রাহ্য করতে পারে।
28	WINCH	উইন্ডো চেঞ্জ(Window change)। যখন কোনো উইন্ডো রিসাইজ করা হয় তখন সিস্টেম এই সিগন্যালটি প্রসেসকে পাঠায়। বিভিন্ন প্রোগ্রাম যেমন top বা less এই সিগন্যালে সাড়া দেয় এবং নিজেদের রিসাইজ করে নতুন উইন্ডো সাইজে নিয়ে আসে।

`kill -1` কমান্ড দিয়ে আপনি চাইলে সকল সিগন্যালের লিস্টটি পেতে পারেন।

এবার আসুন kill কমান্ডের কিছু ব্যবহার দেখি:

```
e@howtocode-pc:~$ xlogo &
[1] 11810
me@howtocode-pc:~$ kill -STOP 11810
me@howtocode-pc:~$ jobs
[1]+  Stopped                  xlogo
me@howtocode-pc:~$ kill -CONT %1
me@howtocode-pc:~$ jobs
[1]+  Running                  xlogo &
me@howtocode-pc:~$ kill -9 %1
[1]+  Killed                   xlogo
me@howtocode-pc:~$
```

- প্রথম আমরা `xlogo &` কমান্ড দিয়ে xlogo এর একটি ব্যাকগ্রাউন্ড প্রসেস তৈরী করেছি। যার Job ID হচ্ছে 1 এবং PID হচ্ছে 11810।
- প্রথম kill কমান্ডটি ছিল `kill -STOP 11810` এখানে আমরা কমান্ডের অপশন হিসেবে সিগন্যালের নাম ব্যবহার করেছি। STOP সিগন্যাল প্রসেসটিকে স্টপ বা পজ করবে। এবং প্রসেসটিকে প্রকাশ করেছি তার PID দিয়ে।
- এরপর `jobs` কমান্ড দিয়ে আমরা দেখতে পাচ্ছি প্রসেসটি Stopped অবস্থায় আছে।
- এরপর আমরা যে kill কমান্ডটি ব্যবহার করেছি সেটি ছিল `kill -CONT %1`। এখানেও অপশন হিসেবে আমরা সিগন্যালটির নাম ব্যবহার করেছি। এবার আমাদের উদ্দেশ্য স্টপ করা প্রসেসটিকে কনটিনিউ করা। তবে এবার আমরা PID না বরং Job ID ব্যবহার করেছি। প্রসেসটির Job ID ছিল 1 তাই আমরা লিখেছি %1।
- এবার আমরা `jobs` কমান্ড দিয়ে দেখলাম স্টপড প্রসেসটি রানিং দেখাচ্ছে।
- এবার আমরা প্রসেসটি বন্ধ করবো। বন্ধ করতে আমরা KILL সিগন্যালটি ব্যবহার করবো। কিন্তু এবারে আমরা সিগন্যালটিকে নাম না তার সংখ্যায় প্রকাশ করছি। KILL সিগন্যালের সংখ্যা হল 9 এবং আমাদের প্রসেসটি Job ID হল 1 তাই আমাদের কমান্ডটি হল `kill -9 %1`

## killall

**killall** একটি বিশেষ কমান্ড যেটি দিয়ে একজন ইউজার বা একটি প্রোগ্রামের সবগুলো প্রসেসকে একসাথে সিগন্যাল পাঠানো যায়। এর কমান্ডকাঠামো এরকম:

```
killall [-u user] [-signal] name...
```

এখানে ইউজার ও সিগন্যাল দুটোই অপশনাল। kill কমান্ডের মত killall ও ডিফল্টভাবে TERM সিগন্যাল পাঠায়।

নীচের উদাহরণে আমরা `xlogo &` কমান্ড দুবার দিয়ে এর দুটি প্রসেস তৈরী করছি। তারপর `killall xlogo` কমান্ড দিয়ে তাদের একসাথে টার্মিনেট করছি:

```
me@howtocode-pc:~$ xlogo &  
[1] 24760  
me@howtocode-pc:~$ xlogo &  
[2] 24761  
me@howtocode-pc:~$ killall xlogo  
[1]-  Terminated          xlogo  
[2]+  Terminated          xlogo
```

## দ্বিতীয় খন্ড

### কনফিগারেশন ও এনভায়রনমেন্ট

আপনারা যারা প্রথম খন্ড শেষ করেছেন দ্বিতীয় খন্ডে স্বাগতম। লিনাক্স ডেভেলপারদের কাছে জনপ্রিয় তার অন্যতম একটি কারন সম্ভবতঃ এটির কাস্টমাইজিবিলিটি। যেহেতু ক্লোজড সোর্সের ব্যবহার মূল কাজে একদমই হয় না বলা চলে, ভালো ডকুমেন্টেশন থাকে তাই সহজেই আপনার অপারেটিং সিস্টেম আপনার দরকারমত পরিবর্তিত ও পরিমার্জিত করে নিতেই পারেন। এই খন্ডে আমরা জানবো আমাদের এনভায়রনমেন্ট সম্পর্কে এবং এর নিয়ন্ত্রণের কৌশল।

- অধ্যায় - এক: এনভায়রনমেন্ট
- অধ্যায় - দুই: প্রস্পট সম্পাদনা

## অধ্যায় - এক

# এনভায়রনমেন্ট

এনভায়রনমেন্ট নিয়ে আমরা আগেও কথা বলেছি। এনভায়রনমেন্ট হচ্ছে শেল দ্বারা সংরক্ষিত কিছু তথ্য যা আমাদের কাজের পরিবেশ নিয়ন্ত্রণ করে। বেশিরভাগ প্রোগ্রাম কীভাবে কাজ করবে তার জন্যে নিজস্ব কনফিগারেশন ফাইলের ওপর নির্ভর করলেও অনেক প্রোগ্রাম আছে এনভায়রনমেন্ট থেকেই নিজেদের কনফিগার করে।

- **এনভায়রনমেন্টের ভিতরে দেখা:** এনভায়রনমেন্ট ও ডেরিয়েবল সম্পর্কে জানা।
- **যেভাবে এনভায়রনমেন্ট তৈরী করা হয়:** স্টার্টআপ ফাইল ও কার্যকর করার পদ্ধতি সম্পর্কে ধারণা।
- **এনভায়রনমেন্ট পরিবর্তন:** এনভায়রনমেন্ট এডিট সম্পর্কে ধারণা এবং nano টেক্সট এডিটরের প্রাথমিক ব্যবহার।

## এনভায়রনমেন্টের ভিতরে দেখা

আমরা এবার দেখতে চেষ্টা করবো এনভায়রনমেন্ট কী কী নিয়ে তৈরী হয়। কী জিনিস বা কী ধরনের তথ্যই বা সে সংরক্ষণ করে। শেল দুই ধরনের ভেরিয়েবল বা চলক সংরক্ষণ করে। এনভায়রনমেন্ট ভেরিয়েবল এবং শেল ভেরিয়েবল। প্রত্যেকটা ভেরিয়েবল এর একটা নাম থাকে। যেমন USER বা HOSTNAME। এবং তার একটা মান থাকে। এই মানটা অপরিবর্তনশীল নয় কাজেই ভেরিয়েবল। যেমন আমার ইউজারনেম me হলে আমার টার্মিনাল থেকে USER ভেরিয়েবল এর মান জানতে চাইলে বলবে 'me' আবার কোনো ইউজারের নাম bob হলে তাকে bob দেখাবে। শেল ভেরিয়েবল হচ্ছে কিছু ভেরিয়েবল যা শেল এনভায়রনমেন্টে যোগ করে। তাছাড়া বাকী সব এনভায়রনমেন্ট ভেরিয়েবল।

ভেরিয়েবল দেখতে আমরা bash এর সাথে বিল্টইন **set** কমান্ডটি ব্যবহার করতে পারি। যেটি একটু দুর্বোধ্যভাবে সব শেল ভেরিয়েবল ও এনভায়রনমেন্ট ভেরিয়েবল সবই দেখাবে। যেহেতু সেটা অনেক লম্বা ও ক্লান্তিকর তাই খুব দরকার না হলে তা ব্যবহার করা হয় না। আমাদের সাধারণ কাজকর্ম শুধু শেল ভেরিয়েবল ঘিরেই আবর্তিত হয়। তাই আমরা শুধু সেটা দেখার জন্য **printenv** কমান্ডটি ব্যবহার করবো। যেহেতু এই লিস্টটাও কম লম্বা না, নিরীক্ষণের সুবিধার জন্য এর আউটপুটকে পাইপের মাধ্যমে **less** কমান্ডের সাথে দেখবো:

```
me@howtocode-pc:~$ printenv | less
```

ফলাফলে less এর মাধ্যমে এমন একটা স্ক্রলবল ডকুমেন্ট হিসেবে দেখাবে:

```

XDG_VTNR=7
LC_PAPER=en_US.UTF-8
SSH_AGENT_PID=2794
LC_ADDRESS=en_US.UTF-8
XDG_SESSION_ID=c2
SELINUX_INIT=YES
LC_MONETARY=en_US.UTF-8
CLUTTER_IM_MODULE=xim
SESSION=gnome
GIO_LAUNCHED_DESKTOP_FILE_PID=3118
TERM=xterm
VTE_VERSION=3409
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
SSH_AGENT_LAUNCHER=upstart
LC_NUMERIC=en_US.UTF-8
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1001/2711
GNOME_KEYRING_CONTROL=/run/user/1001/keyring-ZCsGoa
USER=me
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01
:
```



এখানে আমরা প্রত্যেক ভেরিয়েবলের নাম ও তার মান জানতে পারছি। যেমন USER নামের ভেরিয়েবলের মান 'me'। আমরা শুধু নির্দিষ্ট ভেরিয়েবলের মানও বের করতে পারি:

```
me@howtocode-pc:~$ printenv USER
me
me@howtocode-pc:~$ echo $USER
me
```

এখানে আমরা দুটো পদ্ধতি দেখলাম। প্রথমে **printenv** এর আর্গুমেন্ট হিসেবে ভেরিয়েবলের নাম দিয়েছি। পরের পদ্ধতিতে `echo**` ব্যবহার করেছি। এবং আর্গুমেন্ট হিসেবে `$USER` দিয়েছি। সাধারণত কোথাও ভেরিয়েবলের মান ব্যবহার করতে হলে তার নামের আগে '\$' চিহ্নসহ ব্যবহার করতে হয়।

এবার কিছু ভেরিয়েবল সম্পর্কে জানা যাক:

ভেরিয়েবল	অর্থ
DISPLAY	আপনি যে ডিসপ্লেতে কাজ করছেন। সাধারণত আমাদের একটাই ডিসপ্লে থাকে। তাই এর নম্বর হয় ":0"।
EDITOR	ডিফল্ট টেক্সট এডিটর।
SHELL	ডিফল্ট শেল প্রোগ্রাম।
HOME	আপনার হোম ডিরেক্টরি।
LANG	ডিফল্ট ভাষা সংক্রান্ত তথ্য।
OLD_PWD	এখন যে ডিরেক্টরিতে আছেন তার আগে যে ডিরেক্টরিতে ছিলেন।
PAGER	পেজার। অর্থাৎ ফাইল পড়তে ব্যবহৃত প্রোগ্রাম। সাধারণত এটা হয় less।
PATH	এক্সিকিউটেবল ফাইলের ডিফল্ট ডিরেক্টরিগুলো এখানে কোলোন দিয়ে আলাদা করে দেয়া থাকে। এর সুবিধা হল এইসব ডিরেক্টরির প্রোগ্রামগুলো ব্যবহার করতে প্রোগ্রামের পুরো প্যাথনেম ব্যবহার না করে শুধু নাম ব্যবহার করলেই হয়।
PS1	প্রম্পট স্ট্রিং ১। আপনার শেল প্রম্পটে কী কী দেখাবে তার এর উপর নির্ভর করে। আমরা পরবর্তীতে এটাকে মোডিফাই করে দেখবো।
PWD	কারেন্ট ওয়ার্কিং ডিরেক্টরি বা বর্তমানে যে ডিরেক্টরিতে আছেন।
TERM	আপনার টার্মিনাল প্রোটোকলের নাম।
TZ	টাইমজোন।
USER	আপনার ইউজারনেম।

## যেভাবে এনভায়রনমেন্ট তৈরী করা হয়

আমরা যখন লগইন করি কম্পিউটারে, bash চালু হয় কিছু কনফিগারেশন ফাইল থেকে ঠিক করে নেয় সকল ইউজারের জন্য এনভায়রনমেন্ট কেমন হবে। এই ফাইলগুলো শুরুতে পড়া হয় বলে এদের স্টার্টআপ ফাইল বলে। এরপর bash ইউজারের হোম ডিরেক্টরিতে থাকা আরও কিছু কনফিগারেশন ফাইল পড়ে। কোন ফাইলের পরে কোন ফাইলটি পড়া হবে তা নির্ভর করে শেল সেশনটি কীরকমের তার উপরে। শেল সেশন দু'রকম হয়: ক) লগইন শেল ও খ) নন-লগইন শেল

লগইন শেলে শুরুতেই ইউজারনেম ও পাসওয়ার্ড জানতে চাওয়া হয়। আর লগইন করতে না হলে, যেমনটা আমরা হরহামেশাই ব্যবহার করছি সেটা হচ্ছে নন-লগইন শেল।

লগইন শেল এখান থেকে এক বা একাধিক কনফিগারেশন ফাইল পড়ে:

ফাইল	যা থাকে
/etc/profile	একটি কনফিগারেশন ফাইল যা সব ইউজারের উপরে কার্যকর হয়। এজন্য একে গ্লোবাল কনফিগারেশন স্ক্রিপ্টও বলা হয়।
~/.bash_profile	ইউজারের নিজস্ব স্টার্টআপ ফাইল যা তার হোম ডিরেক্টরিতে থাকে। গ্লোবাল কনফিগারেশনকে ইউজারের প্রয়োজন অনুযায়ী আরও পরিমার্জিত বা পরিবর্তিত করতে এটি ব্যবহৃত হয়। একই জিনিস দুইটি কনফিগারেশন ফাইলে থাকলে এটির থেকে কার্যকর হয়।
~/.bash_login	যদি ~/.bash_profile ফাইলটি না পাওয়া যায় তাহলে এই স্ক্রিপ্টটি পড়ে।
~/.profile	~/.bash_profile বা ~/.bash_login কোনোটিই যদি না পাওয়া যায় তাহলে এটি পড়া হয়।

অন্যদিকে নন-লগইন শেল মাত্র দুটি ফাইল পড়ে:

ফাইল	যা থাকে
/etc/bash.bashrc	নন লগইন শেলের জন্য গ্লোবাল কনফিগারেশন স্ক্রিপ্ট যা সবার জন্য ব্যবহৃত হয়।
~/.bashrc	ইউজারের নিজস্ব স্টার্টআপ ফাইল

একটা নন-লগইন শেল আসলে লগইন এর পরেই পাওয়া যায়। আপনি যখন গ্রাফিকালি লগইন করেন তখনই আপনার লগইন শেল এর কনফিগারেশন ফাইলগুলো পড়ে কার্যকর করা হয়ে যায়। নন-লগইন শেলগুলো আসলে লগইন শেলের চাইল্ড প্রসেস। তাই লগইন শেলের কনফিগারেশনগুলোও এর উপর কার্যকর হয়।

## কী থাকে কনফিগারেশন ফাইলে?

আপনি যদি `less ~/.bashrc` কমান্ডটি ব্যবহার করেন তাহলে নিজস্ব নন-লগইনশেল কনফিগারেশন ফাইলটি দেখতে পারবেন। আপনার ব্যবহৃত ডিস্ট্রিবিউশনের ধরন অনুযায়ী এর হেরফের হবে। যেমন, আমার Ubuntu Gnome 14.04 এর ক্ষেত্রে শুরুর তিন লাইন এরকম:

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples
```

এর প্রত্যেক লাইনের শুরুতে '#' চিহ্ন আছে। তার অর্থ এগুলো কার্যকর হবে না, এগুলো কমেন্ট। কনফিগারেশন ফাইলের বিভিন্ন অংশের কাজ বোঝাতে কমেন্ট ব্যবহার করা হয়। যেমন আমরা যদি bash history সম্পর্কিত কনফিগারেশন দেখি:

```
# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

# append to the history file, don't overwrite it
shopt -s histappend

# for setting history length see HISTSIZE and HISTFILESIZE in bash(1)
HISTSIZE=1000
HISTFILESIZE=2000
```

প্রথমে দুটি কমেন্ট ও তারপর একটি কমান্ড:

```
# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth
```

এখানে বলা হয়েছে ডুপ্লিকেট কমান্ড ও স্পেস দিয়ে শুরু করা কমান্ড হিস্ট্রিতে রাখা হবে না। এবং তারপরের লাইনে এ সম্পর্কিত আরও অপশন দেখতে bash(1) ম্যানুয়াল দেখতে বলা হয়েছে। সবশেষে HISTCONTROL=ignoreboth কমান্ড দিয়ে কাজটি করা হল। একইভাবে হিস্ট্রিসাইজ ও হিস্ট্রিফাইলসাইজ নির্ধারণ করা হয়েছে। এবং বলা হয়েছে যে হিস্ট্রিফাইল ওভাররাইড না করে এ্যাপেন্ড করতে।

## এনভায়রনমেন্ট পরিবর্তন

আমরা জেনেছি এনভায়রনমেন্ট এরপ প্রয়োজনীয় স্টার্টআপ ফাইলগুলো কোনটি কোথায় থাকে। সেগুলোকে সম্পাদন বা এডিট করলেই আমরা এনভায়রনমেন্টের পরিবর্তন ঘটাতে পারবো।

ডিরেক্টরিকে পাথ(PATH) এ যোগ করতে(পাথ বলতে এক্সিকিউটেবল পাথ বোঝানো হচ্ছে। অর্থাৎ, সিস্টেম এক্সিকিউটেবল প্রোগ্রামগুলো ডিফল্টভাবে কোন কোন ডিরেক্টরিতে খুঁজবে।) এবং অতিরিক্ত এনভায়রনমেন্ট ভেরিয়েবল যোগ করতে .bash\_profile(কোনো কোনো ডিস্ট্রিবিউশন যেমন উবুন্টুতে .profile) ফাইলটি ব্যবহার করুন। বাকি সবকিছু .bashrcতে। আর এমনকিছু যদি থাকে যা সকল ইউজার এর জন্য কার্যকর হবে তাহলে তা /etc ডিরেক্টরির profile এ করতে হবে। আপাতত আমরা .profile বা .bash\_profile এবং .bashrc এরকম ব্যক্তিগত স্টার্টআপ ফাইল এডিট করবো।

### টেক্সট এডিটর - আপাতত যা দরকার

টেক্সট এডিটরের সাথে পরবর্তীতে আরো ভালোভাবে পরিচয় করানো হবে। গ্রাফিক্যাল টেক্সট এডিটর আমরা অনেক ব্যবহার করেছি। তবে বইটা যেহেতু কমান্ডলাইনের উপরে তাই আমরা **nano** নামের একটা কমান্ডলাইন টেক্সট এডিটর ব্যবহার করবো। এটা বেশ সহজ। আয়ত্ত করা মোটেই কঠিন না। **nano** চালু করতে হবে এভাবে:

```
nano file_or_files
```

অর্থাৎ প্রথমে কমান্ড এবং তারপর এক বা একাধিক ফাইলের নাম। ওই ডিরেক্টরিতে ফাইলটি থেকে থাকলে ওই ফাইলটি খুলবে নাহলে নতুন ফাইল তৈরী করে নেবে। আমরা যদি হোম ডিরেক্টরি থেকে এই কমান্ড দিই:

```
me@howtocode-pc:~$ cp .bashrc .bashrc.bak  
me@howtocode-pc:~$ nano .bashrc
```

প্রথমে আমরা .bashrc ফাইলটার একটি ব্যাকআপ ফাইল তৈরী করলাম তারপর nano দিয়ে ফাইলটি খুললাম। তখন স্ক্রীনে এরকম কিছু দেখাবে:

GNU nano 2.2.6

File: .bashrc

```
# ~/.bashrc: executed by bash(1) for non-login shells.
# see /usr/share/doc/bash/examples/startup-files (in the package bash-doc)
# for examples

# If not running interactively, don't do anything
case $- in
    *i*) ;;
    *) return;;
esac

# don't put duplicate lines or lines starting with space in the history.
# See bash(1) for more options
HISTCONTROL=ignoreboth

[ Read 117 lines ]
^G Get Help    ^O WriteOut    ^R Read File   ^Y Prev Page   ^K Cut Text    ^C Cur Pos
^X Exit        ^J Justify     ^W Where Is    ^V Next Page   ^U UnCut Text  ^T To Spell
```

সবার প্রথমে বামে টেক্সট এডিটরের নাম ও ভার্সন এবং মাঝখানে ফাইলের নাম। তারপর ফাইলের কন্টেন্ট। একদম শেষ তিনটি লাইনের প্রথম লাইনে ফাইলে লাইন সংখ্যা লেখা আছে। এখানে বিভিন্ন সময় বিভিন্ন প্রয়োজনীয় তথ্য জানায়। তারপর দুই লাইনে দরকারি কমান্ডগুলো। যেমন: ^G Get Help বা ^O WriteOut। এখানে '^' চিহ্ন দিয়ে কন্ট্রোল(Ctrl) কী বোঝায়। অর্থাৎ ^G অর্থ কন্ট্রোল চেপে g চাপতে হবে। এবং এটা চাপলে হেল্প দেখাবে। তেমনি ^O চাপলে রাইটআউট অর্থাৎ ফাইল সেভ করবে। সকল কমান্ডের লিস্ট ^G চেপে দেখে নিতে পারেন।

এ্যারো কী চেপে আপনি কার্সর যেখানে খুশি নিতে পারেন এবং লিখতে পারেন সাধারণ গ্রাফিক্যাল টেক্সট এডিটরের মতই। এর বাইরে সবচেয়ে গুরুত্বপূর্ণ বোধহয় কাট, কপি ও পেস্ট করা। কিন্তু কাট বা কপি করতে গেলে আগে প্রয়োজনীয় অংশটুকুকে নির্বাচন বা মার্ক করতে হবে। যেখান থেকে মার্ক করতে চান সেখানে ^^ চাপবেন। অর্থাৎ কন্ট্রোল চেপে ^। '^' চিহ্নটি 6 এর উপরে থাকে। অর্থাৎ আপনাকে কন্ট্রোল ও শিফট চেপে 6 চাপতে হবে। এরপর নীচে 'Mark Set' লেখা আসবে। এবার আপনি এ্যারো কী চেপে পছন্দের অংশটুকু সিলেক্ট করতে পারেন। মার্ক করে নিয়ে কপি করতে চাইলে M-^ চাপতে হবে। M হল মেটা বা অল্টার কী(Alt) অর্থাৎ আপনাকে অল্টার ও শিফট চেপে 6 চাপতে হবে। কাট করতে চাইলে ^K চাপতে হবে। এবং পেস্ট করতে চাইলে ^U।

ফাইল সেভ ও ক্লোজ করতে যথাক্রমে ^O এবং ^X চাপতে হবে।

## হাতে কলমে

এবার আমরা স্টার্টআপ ফাইল এডিট করবো। আমরা যা করবো তা হয়ত আগেই করা আছে। তবে দ্বিতীয়বার করলে ক্ষতি নেই। প্রথমে `nano .bashrc` কমান্ড দিয়ে .bashrc ফাইলটি খুলুন আর তার শেষে লিখুন:

```
export HISTSIZE=1000
alias ll='ls -lAh'
```

তারপর ^O চেপে সেভ করুন এবং ^X চেপে ফাইল ক্লোজ করুন।

`export HISTSIZE=1000` দিয়ে আমরা HISTSIZE ভেরিয়েবলের মান 1000 করে দিয়েছি। ফলে কমান্ডহিস্ট্রি শেষ ১হাজারটি সংরক্ষণ করবে। `alias ll='ls -lAh'` দিয়ে আমরা `ls -lAh` কমান্ডটির একটি এলিয়াস বানিয়েছি `ll` নামে। অর্থাৎ `ll` কমান্ডটি দিলে লং লিস্ট দেখাবে হিডেন ফাইল সহ। এবং ফাইলগুলোর সাইজ বাইটের বদলে পড়ারযোগ্য এককে(যেমন 22K বা 1.2G) দেখাবে।

আমরা যে পরিবর্তন করলাম তা কিন্তু এখনো কার্যকর হয়নি। কার্যকর করতে হলে শেলকে বলতে হবে কনফিগারেশন ফাইলটি আবার পড়তে। সেজন্য আমরা এই কমান্ডটি দেবো:

```
me@howtocode-pc:~$ source .bashrc
```

এবার `ll` কমান্ডটি দিয়ে দেখুন।

## অধ্যায় - দুই

# প্রম্পট সম্পাদনা

শেলের প্রম্পট একটি খুবই গুরুত্বপূর্ণ অংশ। লিনাক্সের সবকিছুর মতই এটিতেও অনেককিছু নিজের মত করে নেওয়া যায় যা শেল ব্যবহারের সময় বেশ কাজের হতে পারে। এই অধ্যায়ে আমরা প্রম্পট সম্পর্কে বিশদভাবে জানবো।

- **প্রম্পট কাস্টমাইজেশন:** প্রম্পট ব্যবচ্ছেদ, এস্কেপ ক্যারেটের এবং প্রাথমিক কাস্টমাইজেশন
- **প্রম্পট রঙ করা:** প্রম্পটে রঙের ব্যবহার।
- **কার্সরের অবস্থান পরিবর্তন:** কার্সরের অবস্থান পরিবর্তন ব্যবহার করে প্রম্পট স্টাইলিং।

# প্রম্পট কাস্টমাইজেশন

এখন আমরা দেখবো কীভাবে প্রম্পটকে আরো ব্যবহারযোগ্য করা যায়।

## প্রম্পটের ব্যবচ্ছেদ

একটি সাধারণ প্রম্পট(উবুন্টুতে) দেখতে এমন হয়:

```
me@howtocode-pc:~$
```

আপনার ডিস্ট্রিবিউশনে তা অন্যরকম হতেই পারে। এর বিভিন্ন অংশ কেমন হবে তা প্রথম অধ্যায়ের প্রথম [লেসনে](#) আমরা দেখেছি। এখানে আপনার ইউজার নেম, হোস্টনেম ও ওয়ার্কিং ডিরেক্টরি থাকে। এবং সব শেষে থাকে শেলের ধরন(সুপার ইউজার না নরমাল ইউজারের শেল)।

এখন ব্যাপার হচ্ছে প্রম্পট কেমন হবে তা কীভাবে ঠিক করে কম্পিউটার? আপনার ব্যক্তিগত স্টার্টআপ ফাইল .bashrcতে একটা ভেরিয়েবল থাকে **PS1**(Prompt string one) নামে। এটিই নির্ধারন করে প্রম্পটটি কেমন হবে। আপনি PS1 এর মান অবশ্যই .bashrc ফাইলে গিয়ে দেখতে পারেন। তাছাড়াও echo কমান্ড দিয়ে দেখতে পারেন এভাবে:

```
me@howtocode-pc:~$ echo $PS1
\u@\h:\w\ $
```

এর মধ্যে @, :, \$ চিহ্নগুলো তো আমরা প্রম্পটে দেখতেই পাচ্ছি বাকিগুলো পাচ্ছি না। এবং অধিকাংশের সামনেই \"(ব্যাকস্ল্যাশ) আছে। সামনে ব্যাকস্ল্যাশ থাকলে তাদের বলে এস্কেপ ক্যারেক্টার। প্রম্পটের কাছে এরকম বিভিন্ন এস্কেপ ক্যারেক্টারের ভিন্ন মানে আছে। আসুন প্রম্পটে ব্যবহৃত এস্কেপ ক্যারেক্টারের মানে জেনে নিই:



এস্কেপ ক্যারেটর	অর্থ
\a	অ্যাসকি বেল(ASCII bell)। এটা দিলে বীপ করে শব্দ হয়।
\d	বর্তমান দিন। বার, মাস, তারিখ এভাবে সাজানো। যেমন: Sun July 18।
\h	ডোমেইন নেম ছাড়া কম্পিউটারের হোস্টনেম।
\H	সম্পূর্ণ হোস্টনেম।
\j	বর্তমান শেলে চালু থাকা জবের সংখ্যা।
\l	টার্মিনাল ডিভাইসের নাম।
\n	নতুন লাইন তৈরী করে।
\r	ক্যারিজ রিটার্ন। নতুন লাইনের শুরুতে আসে।
\s	শেল প্রোগ্রামের নাম।
\t	বর্তমান সময়। ২৪ঘন্টার হিসেবে। লেখা হয় এভাবে: ঘন্টা:মিনিট:সেকেন্ড।
\T	বর্তমান সময় ১২ঘন্টার হিসেবে।
\@	বর্তমান সময় ১২ঘন্টার হিসেবে AM/PM সহ।
\A	বর্তমান সময় ২৪ঘন্টার হিসেবে কিন্তু সেকেন্ড বাদে।
\u	বর্তমান ইউজারের ইউজারনেম।
\v	শেলের ভার্শন নাম্বার।
\V	শেলের ভার্শন এবং রিলিজ নাম্বার।
\w	বর্তমান ওয়ার্কিং ডিরেক্টরি।
\W	বর্তমান ওয়ার্কিং ডিরেক্টরির শেষাংশ।
!	বর্তমান কমান্ডের হিস্ট্রি নাম্বার।
#	এই শেল সেশনে ব্যবহৃত কমান্ড সংখ্যা।
\$	সুপারইউজার হলে # চিহ্ন দেখাবে, নাহলে \$ চিহ্ন।
[	একাধিক ননপ্রিন্টিং ক্যারেটর শুরুর ব্র্যাকেট।
]	একাধিক ননপ্রিন্টিং ক্যারেটর শেষের ব্র্যাকেট।

এবার হয়ত আপনার কাছে সব পরিষ্কার। আমাদের প্রস্পটে ৪টি এস্কেপ ক্যারেটর ছিল। এগুলো হল:

- \u: এটি দিয়ে ইউজারনেম বোঝানো হয় তাই আপনার ইউজারনেম দেখায়।
- \h: এটি দিয়ে হোস্টনেম।
- \W: এটি বর্তমান ওয়ার্কিং ডিরেক্টরির শেষাংশের জন্য। এর বদলে \w ব্যবহার করা যেত কিন্তু তাতে পুরো পথ দেখাতো বলে স্ক্রীনে অনেকটা জায়গা নষ্ট হত।
- \$: এটি শেলের ধরনের জন্য।

## হাতে কলমে

এবার আমরা প্রম্পট নিয়ে কিছুক্ষণ পরীক্ষা চালাবো। এতে বারোটা বাজতেই পারে প্রম্পটের। তারজন্য আগে আমরা প্রম্পটটির একটি কপি রাখবো:

```
me@howtocode-pc:~$ ps1_old="$PS1"
me@howtocode-pc:~$ echo $ps1_old
\u@\h:\w\$
```

আমরা PS1 এর মান ps1\_old এ কপি করে রাখলাম। এটা হয়েছে নিশ্চিত হতে আমরা `echo $ps1_old` দিয়েছি। আমরা যদি পরবর্তীতে যেকোনো সময় পুরনো প্রম্পট ফিরে পেতে চাই তা করতে পারি এভাবে: `PS1 = "$ps1_old" |`

এবার শুরু করা যাক। প্রথমে এটা:

```
me@howtocode-pc:~$ PS1=
PS1=
```

আমরা প্রম্পটের কোনো মানই দিইনি। যার ফলে পরের লাইনে PS1= দেখিয়েছে। তার পরের লাইনে প্রম্পট আছে কিন্তু এর কোনো লেখা নেই। এবার এটা চেষ্টা করি:

```
PS1="\$ "
PS1="\$ "
$
```

পরের লাইনে যথারীতি PS1 এর মান দেখিয়েছে। কিন্তু তারপরের লাইনে এবার আমরা প্রম্পট দেখতে পাচ্ছি। যা কিনা শুধুমাত্র \$ চিহ্ন। যদি আমরা চাই যে প্রত্যেকবার প্রম্পট আসার সময় একটা বেল দেবে সেটা করতে পারি এভাবে:

```
$ PS1="\[\a\]\$ "
$
```

টার্মিনালে অডিও বেল এনাবেল করা থাকলে এখন থেকে বীপ শব্দ শোনা যাবে। এখন প্রশ্ন হল [ এবং ] কেন ব্যবহার করা হল। এদুটি দিয়ে আমরা \a কে আবদ্ধ করেছি কারন \a ননপ্রিন্টিং ক্যারেক্টার। অর্থাৎ এটি প্রিন্ট হবে না স্ক্রীনে এবং কার্সর সরবে না। প্রম্পটে এরকম কিছু লিখে এই ননপ্রিন্টিং ব্রাকেট দিয়ে আবদ্ধ করতে হয়। এবার একটি তথ্যসমৃদ্ধ প্রম্পট বানাতে চেষ্টা করি যা আসলে কাজে লাগবে:

```
$ PS1="\A \h \$ "
09:57 howtocode-pc $
```

সময় কখনো কখনো কাজে লাগতে পারে যখন একাধিক কাজ কোনটা কখন করছি সেটা জেনে রাখা দরকার হয়। এবার আমরা মোটামুটি কাজের একটা প্রম্পট বানাবো:

```
9:57 howtocode-pc $ PS1="\@ \u@\h->[\w]{\!}\$ "
10:03 AM me@howtocode-pc->[~]{86}$
```

আমরা প্রথম \@ দিয়ে AM/PM এর হিসেবে সময় দিয়েছি। তারপর আমাদের ইউজারনেম এবং হোস্টনেম @ যার মাঝখানে। তারপর -> দিয়ে [] এর মধ্যে \W দিয়ে বর্তমান ওয়ার্কিং ডিরেক্টরির শেষাংশ এবং {} এর মধ্যে ! দিয়ে হিস্ট্রি নম্বর। এবং সবার শেষে \$ দিয়ে শেল মোড।

## প্রম্পট সেড করা

এটা একদমই সহজ কাজ। প্রম্পট সেট করার জন্য যে কমান্ডটি আমরা দিয়েছি .bashrc ফাইলের একদম শেষে সেটি যোগ করে দিলে সবসময়ই এরকম প্রম্পট পাওয়া যাবে। আমরা টেক্সট এডিটরে ফাইলটি খুলবো এবং কমান্ডটি একদম শেষে লিখে সেড দেবো।

এখনি আমরা সেড করা প্রম্পটের ফলাফল দেখতে চাইলে আমাদের `source ~/.bashrc` কমান্ডটি দিতে হবে।

## রঙ যোগ করা

সাদাকালো প্রস্পট কতক্ষণ ভালোলাগে? এবার এটাকে রঙ করা যাক। আমরা কিছু ননপ্রিন্টিং ক্যারেঙ্টার ব্যবহার করে নির্দিষ্ট কিছু রঙের মধ্য থেকে বেছে নিতে পারি। টার্মিনালের টেক্সটে দুইরকম রঙ ব্যবহার করা যায়। ব্যাকগ্রাউন্ড কালার অর্থাৎ টেক্সটের পিছনের রঙ আর টেক্সট কালার বা লেখার রঙ। যেহেতু কালারকোডগুলো ননপ্রিন্টিং ক্যারেঙ্টার তাই এদের [ এবং ] দিয়ে আবদ্ধ করতে হয়।

এবার দেখা যাক একটি কালারকোড। কালো রঙের কালার কোড হল:

```
\033[0;30m
```

এর প্রত্যেকটি অংশের নিশ্চয়ই অর্থ আছে। আসুন দেখে নেয়া যাক:

- **\033**: এটা দিয়ে এক্সেপ সিকুয়েন্স শুরু হয়। কীবোর্ডে এক্সেপ চাপলে এই কোডই পাঠায় কম্পিউটারকে।
- **[ এবং ;**: [ এবং ; এখানে ফরম্যাটিংয়ের কাজে ব্যবহৃত। কোডের তিনটি অংশ যেন মিশে না যায় সেজন্য।
- **0**: এখানে মান হয় 0 হবে অথবা 1। 0 হলে সাধারণ রঙ বা নরমাল কালার। আর 1 হলে বোল্ড কালার বা উজ্জ্বল রঙ।
- **30m**: এটি নির্ধারণ করবে কোন রঙ হবে। যেমন 30m কালো রঙ বুঝায়।

## টেক্সট কালার

আমরা মোট আটটি সাধারণ ও আটটি উজ্জ্বল রঙ ব্যবহার করতে পারি টেক্সট কালারের জন্য।

সাধারণ রঙ:

কোড	রঙ
\033[0;30m	কালো
\033[0;31m	লাল
\033[0;32m	সবুজ
\033[0;33m	বাদামি
\033[0;34m	নীল
\033[0;35m	বেগুনী
\033[0;36m	সবুজাভ নীল বা সাইয়ান(Cyan)
\033[0;37m	হালকা ধূসর

উজ্জ্বল রঙ:

কোড	রঙ
\033[1;30m	গাঢ় ধূসর
\033[1;31m	হালকা লাল
\033[1;32m	হালকা সবুজ
\033[1;33m	হলুদ
\033[1;34m	হালকা নীল
\033[1;35m	হালকা বেগুনী
\033[1;36m	হালকা সায়ান
\033[1;37m	সাদা

## ব্যাকগ্রাউন্ড কালার

ব্যাকগ্রাউন্ড কালারে কোনো উজ্জ্বল রঙ নেই। তাই মোট আটটি রঙ। এর কোড হচ্ছে:

কোড	রঙ
\033[0;40m	কালো
\033[0;41m	লাল
\033[0;42m	সবুজ
\033[0;43m	বাদামি
\033[0;44m	নীল
\033[0;45m	বেগুনী
\033[0;46m	সবুজাভ নীল বা সায়ান(Cyan)
\033[0;47m	হালকা ধূসর

এর বাইরেও একটি কালারকোড আছে যেটি হল \033[0m যা ডিফল্ট রঙে ফেরত আনে।

এবার হাতে কলমে এর ব্যবহার দেখা যাক। গত লেসন(2.2.1.customize.md) আমাদের সর্বশেষ প্রম্পট ছিল:

```
PS1="\@ \u@\h->[\W]{\!}\$ "
```

যদি এটিকে আমরা উজ্জ্বল লাল রঙ দিতে চাই তাহলে আমাদের \033[1;31m ব্যবহার করতে হবে। এবং এটি ননপ্রিন্টিং ক্যারেক্টার বলে [ ও ] দ্বারা আবদ্ধ করতে হবে। অর্থাৎ এটা হবে [\033[1;31m]। এবার তাহলে চেষ্টা করা যাক:

```
01:02 PM utsob@Codex-Zigus->[~]{94}$ PS1="\[\033[1;31m\]\@ \u@\h->[\W]{\!}\$ "
01:03 PM utsob@Codex-Zigus->[~]{95}$
```

এবার আপনি দেখতে পারবেন প্রম্পট লাল হয়ে গেছে। একইভাবে আমরা ব্যাকগ্রাউন্ড কালারও চেঞ্জ করতে পারি। তবে এখনো আমাদের একটি সমস্যা আছে। এরপরের সব লেখাও লাল হয়ে যাচ্ছে। অর্থাৎ আমাদের ঠিক করে দিতে হবে কোনপর্যন্ত রঙ হবে। আমরা প্রম্পটের শেষে যদি \033[0m ব্যবহার করি তাহলে ডিফল্ট রঙে ফিরে আসবে:

```
01:03 PM utsob@Codex-Zigus->[~]{95}$ PS1="\[\033[1;31m\]\@ \u@h->[\w]{\!}\$[\033[0m\] "
01:07 PM utsob@Codex-Zigus->[~]{96}$
```

আমরা আসলে প্রম্পটে একাধিক রঙ ব্যবহার করতেই পারি। আমাদের এই প্রম্পটিকেই আমরা অনেকগুলো রঙ দেবো এভাবে:

```
01:07 PM utsob@Codex-Zigus->[~]{96}$ PS1="\[\033[1;36m\]\@ \[\033[1;31m\]\u\[\033[1;37m\]
01:12 PM utsob@Codex-Zigus->[~]{97}$
```

রীতিমত ভয়ংকর দেখাচ্ছে PS1 ভেরিয়েবলটি। তাইনা? আসুন ভেঙে দেখি:

কোড	অর্থ
<code>\</code> <code>[\033[1;36m\]</code>	হালকা সায়ান রঙের কোড ।
<code>\@</code>	AM/PM এর হিসেবে সময় । এর আগে সায়ান রঙের কোড থাকায় এটি হালকা সায়ান রঙে আসবে ।
<code>\</code> <code>[\033[1;31m\]</code>	হালকা লাল রঙের কোড ।
<code>\u</code>	ইউজারনেম । আগে হালকা লালের কোড থাকায় এটি হালকা লাল রঙে আসবে ।
<code>\</code> <code>[\033[1;37m\]</code>	সাদা রঙের কোড ।
<code>@</code>	'@' চিহ্ন । আগে সাদা রঙের কোড থাকায় এটি সাদা রঙে আসবে ।
<code>\</code> <code>[\033[1;34m\]</code>	হালকা নীল রঙের কোড ।
<code>\h</code>	হোস্টনেমের শেষাংশ । আগে হালকা নীলের কোড থাকায় এটি হালকা নীল রঙে আসবে ।
<code>\</code> <code>[\033[1;37m\]</code>	সাদা রঙের কোড
<code>-&gt;</code>	'->' চিহ্ন । আগে সাদা রঙের কোড থাকায় এটি সাদা রঙে আসবে ।
<code>\</code> <code>[\033[1;32m\]</code>	হালকা সবুজ রঙের কোড ।
<code>[\w]</code>	বর্তমান ওয়ার্কিং ডিরেক্টরির শেষাংশ । আগে হালকা সবুজের কোড থাকায় এটি হালকা সবুজ রঙে আসবে ।
<code>\</code> <code>[\033[1;31m\]</code>	হালকা লাল রঙের কোড ।
<code>{\!}</code>	হিস্ট্রি নাম্বার । আগে হালকা লালের কোড থাকায় এটি হালকা লাল রঙে আসবে ।
<code>\</code> <code>[\033[1;34m\]</code>	হালকা নীল রঙের কোড ।
<code>\\$</code>	শেল মোড । আগে হালকা নীলের কোড থাকায় এটি হালকা নীল রঙে আসবে ।
<code>\[\033[0m\]</code>	ডিফল্ট রঙে ফিরে যাওয়ার কোড

## কার্সরের স্থান পরিবর্তন

**আগের** লেসনে আমরা দেখেছি এক্সেপ ক্যারেটর ব্যবহার করে কীভাবে রঙ দিতে হয় প্রম্পটে। এক্সেপ ক্যারেটর ব্যবহার করে আরো বেশকিছু কাজ করা যায়। তার মধ্যে একটা হল কার্সরের স্থান পরিবর্তন। প্রম্পট ছাড়া অন্য কোনো স্থানে যেমন টার্মিনালের সবচেয়ে উপরের বা নীচের লাইনে কোনোকিছু দেখাতে হলে(যেমন, ঘড়ি বা অন্যকোন তথ্য।) এর প্রয়োজন পড়ে।

টার্মিনাল স্ক্রীনে স্থান নির্দেশ করা হয় গ্রাফের মত করেই। এবং একটি অবস্থান একটি অক্ষর লেখার মত জায়গা দেয়। সবচেয়ে উপরের লাইনের(line) প্রথম অক্ষরের(Column) অবস্থানকে আমরা (0,0) অবস্থান বলতে পারি। সেভাবেই তার ঠিক ডানপাশে (0,1) এবং নীচে (1,0)। এই পদ্ধতিতে আমরা স্ক্রীনে যেকোনো অবস্থান নির্দেশ করতে পারি।

এবার দেখে নেওয়া যাক কার্সরের অবস্থান পরিবর্তনের এক্সেপ কোডগুলো:

কোড	কার্যকারিতা
<code>\033[1;CH</code>	l-তম লাইনের c-তম অক্ষরে কার্সর যাবে। যেমন <code>\033[2;3H</code> লিখলে ৩য় লাইনের ৪র্থ অক্ষরের অবস্থানে যাবে। মনে রাখতে হবে গোনা শুরু হয় ০ দিয়ে।
<code>\033[nA</code>	n লাইন উপরে যাবে।
<code>\033[nB</code>	n লাইন নীচে যাবে।
<code>\033[nC</code>	n অক্ষর সামনে যাবে।
<code>\033[nD</code>	n অক্ষর পিছনে যাবে।
<code>\033[2J</code>	স্ক্রীনের সবকিছু মুছে দেবে এবং কার্সর (0,0) অবস্থানে নেবে।
<code>\033[K</code>	কার্সরের অবস্থান থেকে লাইনের শেষপর্যন্ত মুছে ফেলবে।
<code>\033[s</code>	কার্সরের বর্তমান অবস্থানকে স্মৃতিতে রাখবে।
<code>\033[u</code>	স্মৃতিতে রাখা কার্সরের অবস্থান মনে করবে।

এবার আমরা একটা প্রম্পট তৈরী করবো যা প্রত্যেকবার দেখানোর সময় লাল ব্যাকগ্রাউন্ডের ওপর হলুদ লেখায় একটা ঘড়ি দেখাবে টার্মিনালের প্রথম লাইনে। এর ব্যবহারিক সুবিধা নগণ্য আপনি হয়ত আমাদের পূর্ববর্তী প্রম্পটটি বেশি কাজের মনে করবেন। তাহলে দেখা যাক:

```
PS1="\[\033[s\033[0;0H\033[0;41m\033[K\033[1;33m\t\033[0m\033[u\]<\u@\h \w>\$ "
```

আসুন, এটি ভেঙে দেখা যাক:



কোড	অর্থ
[	ননপ্রিন্টিং কোডের শুরু। এটা দেওয়ার কারন হল এগুলোসহ অবস্থান হিসেব করলে তা নিখুঁত হবে না।
\033[s	কার্সরের বর্তমান পজিশন স্মৃতিতে রাখবে।
\033[0;0H	কার্সরকে (0,0) অর্থাৎ প্রথম লাইনের প্রথম অক্ষরের জায়গায় নিয়ে যাবে।
\033[0;41m	লাল ব্যাকগ্রাউন্ড কালার নেবে।
\033[K	লাইনের শেষ পর্যন্ত সব লেখা মুছে দেবে। আমাদের ব্যাকগ্রাউন্ড কালার লাল বলে একটি লাল বার তৈরী হবে। উল্লেখ্য, কার্সর কিন্তু তার নিজের অবস্থানেই থাকবে।
\033[1;33m	লেখার রঙ হলুদ নির্বাচন করা হবে।
\t	সময় দেখাবে।
\033[0m	টেক্সট ও ব্যাকগ্রাউন্ড কালারকে ডিফল্ট কালারে নিয়ে আসবে।
\033[u	কার্সরকে তার স্মৃতিতে রাখা অবস্থানে নিয়ে যাবে।
]	ননপ্রিন্টিং কোড শেষ।
<\u@\h \W>\$	প্রম্পট দেখাবে।

## তৃতীয় খন্ড

### আটপোরে কমান্ডলাইন

সময়ের সাথে সাথে লিনাক্সের ব্যবহার সাধারণ ব্যবহারকারীদের মধ্যে বেড়েছে। তাই সার্ভারের ব্যবহারের বাইরে এর দৈনন্দিন কাজের উপযোগী প্রোগ্রামও এখন প্রচুর। তাই চাইলে রোজকার কাজের অনেককিছুই কমান্ডলাইনে করা সম্ভব। এই খন্ডের সমস্তকিছু রোজকার ব্যবহারকে মাথায় রেখেই।

- অধ্যায় - এক: প্যাকেজ ম্যানেজমেন্ট
- অধ্যায় - দুই: টেক্সট এডিটর
- অধ্যায় - তিন: স্টোরেজ মিডিয়া
- [অধ্যায় - চার: নেটওয়ার্কিং](3.4.0.networking.md)
- [অধ্যায় - পাঁচ: ফাইল সার্চ](3.5.0.0.search.md)
- অধ্যায় - ছয়: আর্কাইভ ও ব্যাকআপ
- অধ্যায় - সাত: আটপোরে টুলস
- অধ্যায় - আট: প্রোগ্রাম কম্পাইলেশন

## অধ্যায় - এক

# প্যাকেজ ম্যানেজমেন্ট

জিএনইউ/লিনাক্সের বৈচিত্র্যময় জগতের সমৃদ্ধির মূলে সম্ভবত শেয়ারিং। এখানে আপনি অধিকাংশ জিনিস আপনি বিনামূল্যে পাবেন। আর পাবেন স্বাধীনতা। সেটিকে নিজের মত তৈরী করে নিতে এবং আরো মানুষের সাথে শেয়ার করতে। তাই দেখা যায় এমন হতে পারে অনেককিছুই আপনি আগে থেকেই পেয়ে যাচ্ছেন, তৈরী করতে হচ্ছে না। লিনাক্স সিস্টেমে তাই শেয়ার্ড ফাইল বহুলাংশে ব্যবহৃত হয়। এই লাইব্রেরী খাটনি ও খরচ দুটোই কমায়। যে সফটওয়্যার তৈরী করেছে তার যেমন আগেই তৈরী হয়েছে এমনকিছু পুনরায় তৈরী করতে হয়না এজন্য খাটনি কমে তেমনি সফটওয়্যারের আকার হয় ছোট। কারন একই লাইব্রেরী অনেকে ব্যবহার করে।

বুঝতেই পারছেন তাই অনেক সফটওয়্যারই অন্য সফটওয়্যার বা লাইব্রেরীর উপর নির্ভরশীল। তাই সেটিকে ব্যবহার করতে গেলে আরো অনেককিছু ইন্সটল করতে হতে পারে। একসময় যা ছিল একটি গুরুতর সমস্যা। বাস্তবিকই আপনি কতগুলো সফটওয়্যারের নাম মনে রাখতে পারবেন যেগুলো হয়ত আপাতদৃষ্টিতে আপনার কাজে লাগবে না? অথচ লিনাক্স সিস্টেম হাজার হাজার ছোট-বড় সফটওয়্যারের সমন্বয়ে তৈরী হয়।

এই সমস্যার সমাধান হচ্ছে প্যাকেজ ম্যানেজার। প্যাকেজ ম্যানেজার হিসেব রাখে কোথায় প্রয়োজনীয় সফটওয়্যারের প্যাকেজ পাওয়া যাবে, সেটির সাথে আর কী কী লাগবে। হিসেব রাখে সিস্টেম কোন সফটওয়্যারের কোন ভার্সন ইন্সটল করা আছে।

এই অধ্যায়ে আমরা প্রধানদুটি প্যাকেজ ম্যানেজমেন্ট সিস্টেম নিয়ে কথা বলবো। একটি ডেবিয়ান ও ডেবিয়ান এর ফর্করা ব্যবহার করে। অন্যটি রেডহ্যাট ও ফেডোরা ব্যবহার করে।

এবার প্যাকেজ ম্যানেজমেন্ট সম্পর্কিত কিছু তথ্য জেনে নেবো:

## প্যাকেজ

প্রত্যেক সফটওয়্যার বা লাইব্রেরী বিশেষভাবে তৈরী করে রাখা হয় যেন প্যাকেজ ম্যানেজার তা সঠিকভাবে ব্যবহার করতে থাকে। প্রত্যেক প্যাকেজে সফটওয়্যারটি ছাড়াও কিছু তথ্য দেয়া থাকে যাকে মেটাডাটা বলে। তারমধ্যে সফটওয়্যারের নাম, বর্ণনা, এবং এর ডিপেন্ডেন্সি(কোন কোন সফটওয়্যার বা লাইব্রেরীর উপর এটি নির্ভরশীল।) বলা থাকে। বিভিন্ন প্যাকেজ ম্যানেজার বিভিন্ন প্যাকে ব্যবহার করে। যেমন: ডেবিয়ানের প্যাকেজ ম্যানেজারসমূহ ডেব ফাইল ব্যবহার করে যার শেষে এক্সটেনশন হয় .deb আবার rpm ব্যবহার করে আরপিএম ফাইল যার এক্সটেনশন .rpm।

## রিপোজিটরি

রিপোজিটরি হল প্যাকেজের ভাণ্ডার। রিপোজিটরি বিভিন্নরকমের হতে পারে। যেমন সিডিতে বা ব্যক্তিগত স্টোরেজে রাখা রিপোজিটরি। কিন্তু সবচেয়ে বহুল ব্যবহৃত হল অনলাইন রিপোজিটরি। এই অনলাইন রিপোজিটরিগুলো লিনাক্স ডিস্ট্রিবিউশন পরিচালনাকারী প্রতিষ্ঠানগুলো পরিচালনা করে। প্রায় সব ডিস্ট্রিবিউশনেরই নিজস্ব রিপোজিটরি থাকে।

প্যাকেজ ম্যানেজার দিয়ে রিপোজিটরিতে থাকা যেকোনো প্যাকেজ ডিপেন্ডেন্সিসহ ইন্সটল করা হয়। রিপোজিটরিতে একটি ফাইল থাকে যেখানে রিপোজিটরিতে থাকা সব প্যাকেজ ও এর ডার্নন সম্পর্কিত তথ্য থাকে। প্যাকেজ ম্যানেজার এই ফাইলটি পড়ে সোর্স আপডেট করে অর্থাৎ জেনে নেয় ওই রিপোজিটরিতে কী কী প্যাকেজ আছে।

## ডিপেন্ডেন্সি

ডিপেন্ডেন্সি সম্পর্কে আমরা প্যাকেজ সম্পর্কে কথা বলার সময় বলেছি। কোনো সফটওয়্যার অন্য যেসব লাইব্রেরীর উপর নির্ভরশীল সেগুলো ওই সফটওয়্যারের ডিপেন্ডেন্সি।

## উচ্চ ও নিম্নস্তরের প্যাকেজ টুল

প্যাকেজ ম্যানেজমেন্ট ব্যবস্থায় সাধারণত দুইধরনের টুল থাকে। নিম্নস্তরের টুল যেগুলো ইন্সটল বা রিমুভের কাজ করে এবং উচ্চস্তরের টুল যা প্যাকেজ খোঁজা এবং ডিপেন্ডেন্সি সমাধান করে।

আমরা যে দুইধরনের প্যাকেজ ম্যানেজমেন্ট সিস্টেম নিয়ে কথা বলবো তাদের উচ্চ ও নিম্নস্তরের টুলগুলো জেনে নেয়া যাক:

ডিস্ট্রিবিউশন	নিম্নস্তরের টুল	উচ্চস্তরের টুল
ডেবিয়ান, ডেবিয়ানের ফর্কসমূহ যেমন: উবুন্টু, লিনাক্স মিন্ট ইত্যাদি।	dpkg	apt-get, aptitude
ফেডোরা, সেন্টওএস, রেডহ্যাট এন্টারপ্রাইজ লিনাক্স	rpm	yum

## কীভাবে প্যাকেজ ম্যানেজমেন্ট সিস্টেম কাজ করে?

প্যাকেজ ম্যানেজারকে আগে জানতে হয় কোথায় কোথায় সে প্রয়োজনীয় প্যাকেজগুলো পেতে পারে। এরজন্য তারকাছে একটি রিপোজিটরির লিস্ট থাকে। সোর্স আপডেট করলে লিস্টের প্রত্যেকটি রিপোজিটরিতে সে প্যাকেজের লিস্ট ফাইলটি খুঁজে নেয়। তখন তার জানা হয়ে যায় রিপোজিটরিতে কোন কোন ফাইলের কোন কোন ডার্নন আছে। তাছাড়া তার নিজের একটি ডাটাবেজ আছে যেখানে সিস্টেমে ইন্সটল করা সব প্যাকেজের লিস্ট থাকে।

প্যাকেজ ম্যানেজার এরপর প্রস্তুত। তাকে প্যাকেজ খুঁজতে বললে সে এখন খুঁজতে পারে। রিমুভ করতে বললে সে প্যাকেজটি রিমুভ করতে পারে। আবার ইন্সটল করতে বললে আগে সে প্যাকেজের মেটাডাটা পড়ে, জেনে নেয় এর ডিপেন্ডেন্সি গুলো। তারপর মোট কী কী ইন্সটল করতে হবে, তাতে কতটুকু ডাটা খরচ হবে এসব তথ্য উজারকে জানিয়ে দেয়। ইউজার সন্মতি দিলে সে ইন্সটল করে। একইভাবে সে সিস্টেমে যেসব প্যাকেজ পুরনো ডার্ননে আছে, নতুন ডার্নন পেলে আপডেট করে নিতেও সক্ষম।

## প্যাকেজ ম্যানেজারের সাধারণ ব্যবহার

### সোর্স আপডেট

রেডহ্যাট ও ফেডরা অর্থাৎ যারা yum ব্যবহার করে তাদের সোর্স স্বয়ংক্রিয়ভাবে আপডেট হয়। apt-get ব্যবহারকারীরা সোর্স আপডেট করতে পারেন এই কমান্ড দিয়ে:

```
apt-get update
```

আপনি সাধারণ ইউজার হিসেবে এই কমান্ড দিলে পার্মিশন ডিনাইড দেখাবে। আপনার ডিস্ট্রিবিউশনে যদি রুট এক্যাক্সেস এনাবল করা থাকে তাহলে লিখুন `su` এবং এন্টার দিন, রুট এক্যাক্সেসের পাসওয়ার্ড চাইবে। এটি দিলে আপনি রুট হিসেবে কাজ করতে পারবেন। আবার আপনি যদি উবুন্টু বা মিন্ট বা `sudo` ব্যবহার করে এমন কোনো ডিস্ট্রিবিউশন ব্যবহার করেন তাহলে আপনি কমান্ডের আগে `sudo` লাগাবেন। অর্থাৎ, `sudo apt-get update` এই অধ্যায়ের অধিকাংশ কমান্ডই আপনাকে রুট বা এডমিনিস্ট্রেটর হিসেবে দিতে হবে।

## প্যাকেজ খোঁজা

উচ্চস্তরের টুল ব্যবহার করে আমরা কোনো প্যাকেজ খুঁজতে পারি এভাবে:

ধরন	কমান্ড
ডেবিয়ান	<code>apt-cache search search_string</code>
রেড হ্যাট	<code>yum search search_string</code>

**search\_string** এর জায়গায় আপনি যা খুঁজতে চান লিখবেন।

## প্যাকেজ ইন্সটল

প্যাকেজ ইন্সটলের কমান্ডগুলি হল:

ধরন	কমান্ড
ডেবিয়ান	<code>apt-cache install package_name...</code>
রেড হ্যাট	<code>yum install package_name...</code>

অর্থাৎ, আপনি যদি `vim` ইন্সটল করতে চান তাহলে ডেবিয়ানের ধরনের ডিস্ট্রিবিউশনে লিখবেন: `apt-get install vim` এবং রেড হ্যাট এর ধরনে `yum install vim`। এরপর প্যাকেজ ম্যানেজার আপনাকে হিসেব করে বলে দেবে ডিপেন্ডেন্সিসহ মোট কতটুকু ডাউনলোড করতে হবে এবং সম্মতি চাইবে। আপনি 'y' চাপলে সে ডাউনলোড শুরু করবে এবং ডাউনলোড শেষে ইন্সটল করে নেবে।

## আলাদাভাবে ডাউনলোড করা প্যাকেজ ইন্সটল

আলাদাভাবে ডাউনলোড করা প্যাকেজ আপনি নিম্নস্তরের টুল দিয়ে ইন্সটল করতে পারেন এভাবে:

ধরন	কমান্ড
ডেবিয়ান	<code>dpkg -i package-file</code>
রেড হ্যাট	<code>rpm -i package-file</code>

তবে এ পদ্ধতিতে ডিপেন্ডেন্সির সমস্যা আপনাকে নিজেই সলভ করতে হবে। স্বাভাবিকভাবেই উচ্চস্তরের প্যাকেজ ম্যানেজারই ব্যবহার করা ভালো।

## প্যাকেজ রিমুভ করা

আপনি যেকোনো প্যাকেজ রিমুভ করতে পারেন এভাবে:

ধরন	কমান্ড
ডেবিয়ান	<code>apt-get remove package_name</code>
রেড হ্যাট	<code>yum erase package_name</code>

## সিস্টেম আপডেট করা

আপনি আপনার সিস্টেমের সব প্যাকেজ আপডেট করতে পারেন এভাবে:

ধরন	কমান্ড
ডেবিয়ান	<code>apt-get upgrade</code>
রেড হ্যাট	<code>yum update</code>

এক্ষেত্রেও আপনার কাছে ইন্টলের মতই সম্মতি চাওয়া হবে।

উবুন্টুতে আপনাকে সিস্টেম সম্পূর্ণরূপে আপডেট করতে `apt-get upgrade` এর পর `apt-get dist-upgrade` দিতে হবে।

## ইন্টল করা প্যাকেজের লিস্ট করা

আমরা সিস্টেমে ইন্টল করা সকল প্যাকেজের লিস্ট পেতে পারি এভাবে:

ধরন	কমান্ড
ডেবিয়ান	<code>dpkg --get-selections</code>
রেড হ্যাট	<code>rpm -qa</code>

## প্যাকেজ ইন্টল করা আছে কিনা নিশ্চিত হওয়া

দীর্ঘ লিস্টে আপনার ইন্টল করা প্যাকেজটি আছে কিনা দেখার চেয়ে এই পদ্ধতিতে দেখা সহজ:

ধরন	কমান্ড
ডেবিয়ান	<code>dpkg-query -f='\${Package}' -W -f='\${Package}'</code>
রেড হ্যাট	<code>rpm -q package_name</code>

## ইন্সটল করা প্যাকেজ সম্পর্কে তথ্য জানা

ইন্সটল করা প্যাকেজ সম্পর্কে তথ্য জানতে ব্যবহার করুন:

ধরন	কমান্ড
ডেবিয়ান	<code>apt-cache show package_name</code>
রেড হ্যাট	<code>yum info package_name</code>

## অধ্যায় - দুই

# টেক্সট এডিটর

একজন সাধারণ ব্যবহারকারী, যে কিনা একটু ইন্টারনেট ব্রাউজিং, একটু সিনেমা দেখা, গেম খেলার জন্যই শুধু কম্পিউটার ব্যবহার করে তারকাছে আসলেই টেক্সট এডিটর কোনো গুরুত্বপূর্ণ ব্যাপার না। কিন্তু আপনারা যারা এই টিউটোরিয়াল এতদিন ধরে পড়ছেন, তারা জানেন টেক্সটর গুরুত্ব। আমরা যা করেছি সব লিখেই করেছি। পুরো অপারেটিং সিস্টেম বহু বহু মানুষ লক্ষকোটি লাইন লিখে তৈরী করেছেন। গ্রাফিক্যাল জগতে আমরা gedit, kate উইন্ডোজ ব্যবহারকারীরা notepad এর সাথে পরিচিত। আমরা দেখেছি অনেক IDE(Integrated Development Environment) যার সাথে প্রোগ্রামিং ল্যান্সুয়েজ স্পেসিফিক টেক্সট এডিটর থাকে। যেমন: Codeblocks বা SPE। মজার বিষয় হল আইডিইগুলো এতই জনপ্রিয় যে আমাদের সিএসই স্টুডেন্টরা অনেকে জানেনই না যে আইডিই ছাড়াও প্রোগ্রাম লিখে কম্পাইল করা যায়!

অনেক কারন আছে যার জন্য আপনি এই প্রাচীনপন্থী কমান্ডলাইন টেক্সট এডিটর শিখতে চাইবেন। এমন অবস্থায় পড়তে পারেন তা সার্ভারেই হোক বা আপনার নিজের কম্পিউটারে এমন কোনো সমস্যা যার জন্য গ্রাফিক্যাল এনভায়রনমেন্টে কাজ করতে পারছেন না, তখন এই টেক্সট এডিটরগুলোর একটি আপনাকে বাঁচাতে পারে। তাছাড়াও আপনার দক্ষতা এমন পর্যায়ে আপনি নিয়ে যেতেই পারেন যখন গ্রাফিক্যাল টেক্সট এডিটরের চেয়ে এগুলোই আপনার কাছে সুবিধাজনক মনে হবে।

আমরা এই অধ্যায়ে ৩টি টেক্সট এডিটরের সাথে পরিচিত হবো। ন্যানো(Nano), ভিম(VIM) এবং ইম্যাকস(Emacs)। ন্যানো একারনেই যে এটি খুবই সহজ, এবং এখনকার প্রায় সব ডিস্ট্রিবিউশনের সাথেই দেওয়া থাকে। এর বাংলা সাপোর্ট যথেষ্ট ভালো। VIM ঐতিহ্যগত কারনে। ইউনিক্স সিস্টেমে ভিম একটি বহুল ব্যবহৃত এডিটর। তবে এটিতে বাংলা লেখা যায় না। এটি খুবই ক্ষমতাধর এডিটর। আর ইম্যাকস হচ্ছে ভিমের সবচেয়ে বড় প্রতিদ্বন্দ্বী। তুলনামূলক সহজ। প্রচুর অপশন ও ব্যবহারযোগ্যতা এবং বাংলা সাপোর্ট আছে।

আমি প্রত্যেকটিতেই বাংলা সাপোর্ট সম্পর্কে বলেছি তবে আপনার টার্মিনাল ইমুলেটরের বাংলা সাপোর্ট ভালো না হলে কোনোটাতেই বাংলা লিখতে পারবেন না ঠিকভাবে। এ পর্যন্ত আমি konsole এবং ড্রপডাউন টার্মিনাল yakuake এ ঠিকঠাক বাংলা সাপোর্ট পেয়েছি। আপনারা এই দুটি ইন্সটল করে নিতে পারেন।

আপনাকে তিনটেই শিখতে হবে এমন না। আপনি চাইলে শুধু ন্যানোই যথেষ্ট আবার তার সাথে ভিম ও ইম্যাকস এর একটি শিখতে পারেন। আবার চাইলে তিনটিই।

- **ন্যানো:** ন্যানো এর ব্যবহার।
  - **ন্যানোর প্রাথমিক ব্যবহার:** ন্যানো এডিটরের প্রাথমিক ব্যবহার।
  - **ন্যানো - এডিটিং এবং নেভিগেশন:** এডিটিং, সার্চ এবং রিপ্লেস এবং নেভিগেশন।
  - **ন্যানো কনফিগারেশন:** ন্যানো কনফিগার করা।
- **ভিম:** ভিম এর ব্যবহার।
  - **ভিমের এডিটিং মোড:** ভিমের মোড সম্পর্কিত ধারণা।
  - **ভিম-এর বেসিক এডিটিং:** ভিম-এর বেসিক এডিটিং।
  - **ভিম: সার্চ এ্যান্ড রিপ্লেস:** ভিমে সার্চ এবং রিপ্লেস অপারেশন।



- ভিন্ন: একাধিক ফাইল নিয়ে কাজ করা: ভিন্নে একাধিক ফাইল নিয়ে কাজ করা।
- ইম্যাকস্: ইম্যাকস্ এর ব্যবহার।
  - ইম্যাকস্: প্রথম ধাপ: ইম্যাকস্ চালু, ইন্টারফেস পরিচিতি ও বন্ধ।
  - ইম্যাকস্: ক্যারেটার, কী এবং কমান্ড: ক্যারেটার, কী এবং কমান্ড সম্পর্কিত আলোচনা।
  - ইম্যাকস্: বেসিক এডিটিং: ইম্যাকসে বেসিক এডিটিং।
  - ইম্যাকস্: সার্চ এন্ড রিপ্লেস: ইম্যাকসে সার্চ এবং রিপ্লেস।
  - ইম্যাকস্: একাধিক ফাইল এডিট করা: ইম্যাকসে একাধিক ফাইল নিয়ে কাজ করা।

## ন্যানো (Nano)

ন্যানো সম্পর্কে আমরা আগে কথা বলেছি। আমরা যে তিনটি টেক্সট এডিটর দেখবো এটি তারমধ্যে সবচেয়ে সহজ, ব্যবহারযোগ্যতাও নিতান্ত কম নয়। এমন সম্ভবনা খুবই কম যে আপনার সিস্টেমে ন্যানো আগে থেকেই ইন্সটল করা নেই। ন্যানো আছে কিনা দেখতে `which nano` কমান্ড দিন। যদি না থাকে তবে:

ডেবিয়ান সিস্টেমে: `# apt-get install nano` রেড হ্যাট সিস্টেমে: `# yum install nano` আর্চ লিনাক্স সিস্টেমে: `# pacman -S nano`

ব্যবহার করুন। উল্লেখ্য, শুরু '#' আপনাকে লিখতে হবে না। এর অর্থ এই কমান্ডটি আপনাকে সুপারইউজার হিসেবে দিতে হবে। হয় রুট একাউন্ট থেকে বা `sudo` সহ।

আপনি ন্যানোর জন্য তৈরী!

- **ন্যানোর প্রাথমিক ব্যবহার:** ন্যানো এডিটরের প্রাথমিক ব্যবহার।
- **ন্যানো - এডিটিং এবং নেভিগেশন:** এডিটিং, সার্চ এবং রিপ্লেস এবং নেভিগেশন।
- **ন্যানো কনফিগারেশন:** ন্যানো কনফিগার করা।

# ন্যানোর প্রাথমিক ব্যবহার

## ন্যানো চালু করা

ন্যানো চালু করার কয়েকটি পদ্ধতি আছে। আপনি যদি শুধু `nano` কমান্ডটি দেন তাহলে ন্যানো একটি নামহীন ফাঁকা বাফারসহ চালু হবে। এক্ষেত্রে বাফার (buffer) সম্পর্কে বলে রাখি। অধিকাংশ টেক্সট এডিটরকে যখন কোনো ফাইল খুলতে বলা হয়, সে ওই ফাইলের একটি কপি বা প্রতিলিপি খোলে। সেটাতে কাজ করে সেভ করলে তবেই অরিজিনাল ফাইলকে ওভাররাইট করে। এই সাময়িক কপিকে বাফার বলে। কোনো ফাইলের নাম না দিয়ে তৈরী এই ফাঁকা বাফার সেভ করতে গেলে আপনায় জিজ্ঞাসা করবে কোন ফাইলনেম দিয়ে সেভ করতে চান। তেমনি আপনি আগেই ফাইলের নাম সহ ন্যানো চালু করতে পারেন এভাবে:

```
me@howtocode-pc:~$ nano filepath [filepath2]...
```

অর্থাৎ, **nano** কমান্ডের আর্গুমেন্ট হিসেবে এক বা একাধিক ফাইলের পাথ। যদি ফাইলটি আগে থেকে থাকে তাহলে সেই ফাইলটিই খুলবে। আর যদি না থাকে তো নতুন ফাইল খুলবে।

## ন্যানোর বিভিন্ন অংশ

ন্যানো চালু করার পর আপনি এরকম দেখবেন:

GNU nano 2.2.6

New Buffer

```
^G Get Help  ^O WriteOut  ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit      ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text^T To Spell
```

- প্রথম লাইনে আমরা সাদা একটি বার দেখবো। যার বামপাশে ন্যানোর ভার্সন সহ নাম দেখাবে। এখানে ছিল 'GNU nano 2.2.6'।

- তারপরই ফাইলের নাম। আমরা কোনো আর্গুমেন্ট ছাড়া ন্যানো চালু করেছি তাই আমাদের একটা নতুন বাফার তৈরী করে দিয়েছে যার নাম 'New Buffer'।
- তারপরই আপনি কাসারসহ লেখার জায়গা পাবেন।
- নীচের দিক থেকে উপরে উঠতে থাকলে তৃতীয় লাইনটি মিনিবাফার। এখানে বিভিন্ন গুরুত্বপূর্ণ তথ্য দেখায় এবং বিভিন্ন কমান্ডের অংশ হিসেবে লেখার সুযোগ দেয়।
- নীচের দিক থেকে শেষ দুই লাইনে প্রয়োজনীয় কমান্ডগুলো দেয়া থাকে। ন্যানোর কমান্ডে দুইধরনের সংকেত ব্যবহৃত হয়। '^' চিহ্ন দিয়ে CTRL বোঝায় এবং 'M-' দিয়ে মেটা বা ALT যদি তা কমান্ডের সামনে বসে। অর্থাৎ '^M' মানে কিন্তু CTRL+CTRL না। শুধু প্রথম '^'টা CTRL পরেরটা 'M' চিহ্ন যা সাধারণত শিফট চেপে 6 চাপলে আসে।

## সেড, এক্সিট এবং হেল্প

ন্যানোর নীচে দুই লাইনে দেখে আপনি এই তিনটি কাজ করার উপায় হয়ত এতক্ষণে পেয়ে গেছেন।

### সেড

সেড করতে ^O চাপুন। যদি আগে থেকে ফাইলের নাম না দেয়া থাকে তাহলে ফাইলের নাম জিজ্ঞাসা করবে। আর যদি আগে থেকেই ফাইলের নাম দিয়ে রাখেন সেই নামটিই দেখাবে। আপনি ওই ফাইলের উপরেই সেড করতে চাপলে সরাসরি এন্টার চাপবেন। আর যদি অন্য কোনো নামে সেড করতে চান তাহলে ডিফল্ট নামটি মুছে সেই নামটি লিখে এন্টার দেবেন।

### হেল্প

^G চাপলে একটি হেল্প মেন্যু দেখাবে। যেখানে সকল কমান্ড ও তার কার্যকারিতা পাবেন।

### এক্সিট

ন্যানো বন্ধ করতে হলে ^X চাপতে হবে। যে ফাইলটি নিয়ে কাজ করছেন সেটিতে কিছু করা হয়েছে কিন্তু সেড করা হয়নি এমন হলে আপনাকে জিজ্ঞাসা করবে যে ফাইলটি সেড করবে নাকি সেড না করেই বন্ধ করবে।

## ন্যানো - এডিটিং এবং নেভিগেশন

ন্যানোতে লেখা শেখানোর কিছু নেই। আপনি স্বাভাবিক যেভাবে লেখেন সেভাবেই লিখতে পারেন। ব্যাকস্পেস চেপে লেখা মুছতে পারেন। এ্যারো কী ব্যবহার করে কার্সরের অবস্থান পরিবর্তন করতে পারেন।

### কাট, কপি ও পেস্ট করা

কাট বা কপি করতে হলে আপনাকে আগে প্রয়োজনীয় অংশটুকু মার্ক করতে হবে। '^' অর্থাৎ CTRL-^ চাপলে আপনি নীচে 'Mark set' লেখা দেখবেন তারপর এ্যারো কী চেপে প্রয়োজনীয় অংশটুকু মার্ক করুন। তারপর:

- কপি করতে M-^ চাপুন, অথবা
- কাট করতে ^K চাপুন।

পেস্ট করতে ^U চাপুন।

### সার্চ এবং রিপ্লেস

আপনি আপনার ডকুমেন্টে কোনো একটা লেখা খুঁজতে পারেন। সেটাকে রিপ্লেস করতে পারেন অন্য শব্দ দিয়ে। তারজন্যে আপনাকে আগে ^W চাপতে হবে। এবার মিনিবাফারে 'Search: ' লেখা আসবে। এখানে আপনি যা সার্চ করতে চান লিখবেন। তারপর এন্টার চাপলে সেই লেখাটি থাকলে কার্সরের ঠিক পরেই যেখানে আছে দেখাবে। আবারও একই জিনিস সার্চ করতে হলে ^W চাপলে দেখবে সার্চের পাশে ব্রাকেটের ভিতরে আগের সার্চস্ট্রিংটি আছে। এখন শুধু এন্টার দিলেই হবে।

রিপ্লেস করতে হলে প্রথমে ^W চাপতে হবে। তারপর ^R চাপলে রিপ্লেস মোড সক্রিয় হবে। মিনিবাফারে 'Search (to replace):' লেখা আসবে। এবার যে স্ট্রিংটি রিপ্লেস করতে চান সেটি লিখে এন্টার দেবেন। এবার মিনিবাফারে লেখা আসবে 'Replace with:' এখানে যে স্ট্রিংটি বসাতে চান লিখে এন্টার দেবেন। তখন আপনার সার্চ করা স্ট্রিংটি হাইলাইট করে দেবে এবং মিনিবাফারে জিজ্ঞাসা করবে লেখাটি রিপ্লেস করবে কিনা। আপনি Y চেপে সম্মতি দিতে পারেন বা N চেপে অসম্মতি জানাতে পারেন। সম্মতি দিলে রিপ্লেস করবে এবং সার্চ স্ট্রিংটি পরে যেখানে আছে নিয়ে গিয়ে আবার সম্মতি চাইবে। তাছাড়াও আপনি A চেপে সবগুলো রিপ্লেস করতে পারেন একবারে বা ^C চেপে ক্যানসেল করতে পারেন।

### একাধিক ফাইলের মধ্যে চলাফেরা

আপনি যদি একাধিক ফাইল ন্যানোতে একবারে খুলে থাকেন, তাহলে পরবর্তী ফাইলে যেতে পারেন ^V বা F8 চেপে এবং পূর্ববর্তী ফাইলে যেতে পারেন ^Y বা F7 চেপে।

## ন্যানো কনফিগারেশন

ন্যানোকে কনফিগার করতে হলে আপনার হোমের `.nanorc` নামের ফাইলটি এডিট করতে হবে। প্রশ্ন হল, কেন কনফিগার করবেন? উত্তরটাও সহজ, আপনার সুবিধার্থে। ধরে নিলাম আপনি পাইথনে প্রোগ্রাম লেখেন। আপনার কোড ইন্ডেন্ট করতে হয় ৪টি স্পেসের সমপরিমাণ ট্যাব দিয়ে। সবসময়ই ট্যাবের চেয়ে সরাসরি ৪টি স্পেস ব্যবহার করা ভালো। তো, আপাতত দেখা যাবে দুটি জিনিস হলে আপনার খাটনি অনেকটা কমবে:

- আপনি যখনি ট্যাব চাপবেন, যদি ৪টি করে স্পেস বসিয়ে নেয় তার বদলে।
- যদি কোড স্বয়ংক্রিয়ভাবে ইন্ডেন্ট করে দেয়।

এটা করতে প্রথমে দেখুন আপনার হোমে `.nanorc` ফাইলটি আছে কিনা। না থাকলে তৈরী করে নিন। ফাইলটি খুলুন। তারপর এই লাইনগুলো লিখুন:

```
set autoindent
set tabsize 4
set tabtoospace
```

এবার ফাইলটি সেভ করুন। আমরা তাহলে কী করলাম? প্রথমে **set autoindent** দিয়ে আমরা বলেছি স্বয়ংক্রিয়ভাবে ইন্ডেন্ট করতে। তারপর **set tabsize 4** দিয়ে বলেছি ট্যাব এর দৈর্ঘ্য হবে ৪ ক্যারেটারের সমান। তারপর **set tabtoospace** দিয়ে বলেছি ট্যাব এর বদলে স্পেস ব্যবহার করতে।

এর বাইরেও আপনি অনেককিছু পরিবর্তন করতে পারেন ন্যানোর। বিস্তারিত দেখতে টার্মিনালে লিখুন:

```
man nanorc
```

# ভিম (VIM)

ভিম(VIM: Vi Improved) Vi text editor এর একটি আধুনিক ভার্সন। ভিম খুবই শক্তিশালি এবং এবং রিসোর্সফ্রেন্ডলি এডিটর। অর্থাৎ এটি খুব সামান্য র‍্যাম ও প্রসেসর খরচ করে। কোড এডিটর হিসেবে এটি খুবই জনপ্রিয় এবং অধিকাংশ জিএনইউ/লিনাক্স অপারেটিং সিস্টেম এটি ডিফল্টভাবে দিয়ে রাখে। যদি না থাকে তবে এভাবে ইন্সটল করতে পারেন:

ডেবিয়ান সিস্টেমে: `# apt-get install vim` রেড হ্যাট সিস্টেমে: `# yum install vim` আর্চ লিনাক্স সিস্টেমে: `# pacman -S vim`

এবার শুরু করা যাক!

## ভিম চালু ও বন্ধ করা

ভিম চালু করতে লিখুন:

```
me@howtocode-pc:~$ vim
```

ফলে এরকম একটাকিছু আপনি দেখতে পাবেন:







কী(key)	কাজ
l বা Right Arrow	ডানদিকে একঅক্ষর সরবে।
h বা Left Arrow	বামদিকে একঅক্ষর সরবে।
j বা Down Arrow	নীচের লাইনে যাবে।
k বা Up Arrow	উপরের লাইনে উঠবে।
0	বর্তমান লাইনের শুরুতে যাবে।
^	প্রথম নন-হোয়াইটস্পেস অক্ষরে যাবে।
\$	বর্তমান লাইনের শেষে যাবে।
w	পরবর্তী শব্দ বা যতিচিহ্নের শুরুতে যাবে।
W	পরবর্তী শব্দের শুরুতে যাবে, যতিচিহ্ন আমলে নেবে না।
b	পূর্ববর্তী শব্দ বা যতিচিহ্নের শুরুতে যাবে।
B	পূর্ববর্তী শব্দের শুরুতে যাবে, যতিচিহ্ন আমলে নেবে না।
Ctrl-f বা Page Down	একপৃষ্ঠা নীচে নামবে।
Ctrl-b বা Page Up	একপৃষ্ঠা উপরে উঠবে।
numberG	number এর জায়গায় লেখা লাইনে যাবে। উদাহরণস্বরূপ: 4G মানে চতুর্থ লাইনে যাবে।
G	ফাইলের শেষ লাইনে যাবে।

## ভিম-এর বেসিক এডিটিং

ভিম-এর ক্ষেত্রে একটা জিনিস মনে রাখা দরকার যে ডিফল্টভাবে এটি কমান্ড মোডে চালু হয় কারন এর মূল ব্যবহার কমান্ড মোডেই। ইন্সার্ট মোডে আপনি লেখার পর এক মুহূর্তও থাকবেন না এটা হবে বেস্ট প্রাকটিস। এবার ভিম-এর বেসিক এডিটিং ফিচারগুলো দেখে নেওয়া যাক:

### লেখা

লেখার জন্য আমরা ইতমধ্যে ইন্সার্ট মোডে গিয়েছি 'i' চেপে। তাছাড়া যেকোনো লাইনের একদম শেষে কার্সর নিয়ে ইন্সার্ট মোডে যেতে শুধু 'A' চাপলেই হবে। সাধারণত আমরা লাইনের শেষে আরো লেখা যোগ করি বলে এই ফিচার রাখা হয়েছে। তাছাড়া আপনি 'o' চাপলে যে লাইনে কার্সর আছে তার নীচে বা 'O' চাপলে তার উপরে একটা ফাঁকা লাইন তৈরি করবে এবং কার্সর সেখানে ইন্সার্ট মোডে যাবে।

### লেখা ডিলিট করা

ইন্সার্ট মোডে আপনি ব্যাকস্পেস চেপে লেখা মুছতে পারেন। তাছাড়াও অনেকগুলো কমান্ড আছে যা দ্রুতি এনে দেবে কাজে। আসুন, সেগুলো দেখা যাক:

কমান্ড	কাজ
x	বর্তমানে কার্সরে থাকা অক্ষরটি মুছবে।
5x	বর্তমান অক্ষর ও পরবর্তী চারটি অক্ষর মুছবে।
dd	বর্তমান লাইন মুছবে।
5dd	বর্তমান লাইন ও পরবর্তী ৫ লাইন মুছবে।
dW	বর্তমান কার্সরের অবস্থান থেকে পরবর্তী শব্দের শুরু পর্যন্ত মুছবে।
d\$	বর্তমান কার্সরের অবস্থান থেকে লাইনের শেষ পর্যন্ত মুছবে।
d0	বর্তমান কার্সরের অবস্থান থেকে লাইনের শুরু পর্যন্ত মুছবে।
d^	কার্সরের বর্তমান অবস্থান থেকে পরবর্তী নন-হোয়াইটস্পেস অক্ষর পর্যন্ত মুছবে।
dG	বর্তমান লাইন হতে ফাইলের শেষ পর্যন্ত মুছবে।
d20G	বর্তমান লাইন থেকে ২০তম লাইন পর্যন্ত মুছবে।

উপরের টেবিলে যেসবক্ষেত্রে সংখ্যা ব্যবহার করা হয়েছে সেসবক্ষেত্রে আপনি আপনার প্রয়োজনমত সংখ্যা বসাতে পারেন। যেমন, আপনি 9dd কমান্ড দিলে ৯টি লাইন মুছবে।

### কাট, কপি(ইয়াক) ও পেস্ট

**d** দিয়ে শুরু হওয়া যেসব কমান্ড আমরা আগের টেবিলে দেখলাম সেগুলো আসলে ডিলিট না, কাট করে। একইভাবে আপনি 'y' এর সাথে এরকম কমান্ড তৈরী করতে পারেন যেগুলো কপি করবে। যেমন, **dd** যেমন একলাইন কাট করে 'yy' চেপে আপনি কপি করতে পারেন। আর পেস্ট করতে পছন্দনীয় জায়গায় কার্সর রেখে 'p' চাপুন। এবার কপির কমান্ডগুলো দেখা যাক:

কমান্ড	কাজ
yy	বর্তমান লাইন কপি করবে।
5yy	বর্তমান ও পরবর্তী ৪ লাইন কপি করবে।
yW	বর্তমান কার্সরের অবস্থান থেকে পরের শব্দের শুরু পর্যন্ত কপি করবে।
y\$	বর্তমান কার্সরের অবস্থান থেকে লাইনের শেষ পর্যন্ত কপি করবে।
y0	বর্তমান কার্সরের অবস্থান থেকে লাইনের শুরু পর্যন্ত কপি করবে।
y^	কার্সরের বর্তমান অবস্থান থেকে পরবর্তী নন-হোয়াইটস্পেস অক্ষর পর্যন্ত কপি করবে।
yG	বর্তমান লাইন হতে ফাইলের শেষ পর্যন্ত কপি করবে।
y20G	বর্তমান লাইন হতে ফাইলের ২০তম লাইন পর্যন্ত কপি করবে।

## মার্ক, কপি, কাট, ডিলিট, পেস্ট

মনে মনে হয়ত গালাগালি করছেন ডিমকে। নরমাল এডিটরে কি সুন্দর একটা অংশ সিলেক্ট করে কাট-কপি-পেস্ট-ডিলিট করেন আর এখানে রাজ্যের ঝামেলা! হতাশার কিছু নেই। ডিম সেটারও সুযোগ রেখেছে। আপনাকে যা করতে হবে তা হলো:

- যেখান থেকে কপি করা শুরু করতে চান কমান্ড মোডে সেখানে কার্সর নেবেন।
- তারপর 'v' চাপবেন। যদি লাইন বাই লাইন সিলেক্ট করতে চান তাহলে 'V' চাপবেন।
- এবার এ্যারো কী দিয়ে নির্দিষ্ট অংশ সিলেক্ট করবেন।
- সিলেক্ট করা অংশটুকু ডিলিট বা কাট করতে চাপবেন **d** অথবা কপি করতে **y**।
- তারপর যেখানে পেস্ট করতে চান সেখানে কার্সর নিয়ে চাপুন **p**

## ভিম: সার্চ এ্যান্ড রিপ্লেস

ভিমের সার্চ ও রিপ্লেসের ক্ষমতা প্রশংসনীয়। তারমধ্যে প্রয়োজনীয়টুকু আমরা দেখবো।

### লাইনের মধ্যে সার্চ করা

**f** চেপে আমরা কোনো লাইনের মধ্যে কোনো নির্দিষ্ট অক্ষর খুঁজতে পারি। যেমন `fa` লিখলে লাইনে কার্সরের পরে প্রথম যেখানে 'a' অক্ষরটি আছে সেখানে কার্সর নিয়ে যাবে। আমরা আবার ';' চেপে পরের 'a' এর কাছে কার্সর নিতে পারি।

### ফাইলের মধ্যে সার্চ করা

কোনো ফাইলে নির্দিষ্ট শব্দ বা শব্দগুচ্ছ খুঁজে নিতে '/' ব্যবহার করা হয়। '/' চাপলে নীচে '/' লেখা আসবে। তারপর যেটি খুঁজতে হবে সেটি লিখে এন্টার দিলে কার্সর সেখানে চলে যাবে। তারপরের সার্চ স্ট্রিংটির কাছে যেতে হলে আমরা 'n' চেপে সার্চের পুনরাবৃত্তি করা যেতে পারে।

### গ্লোবাল সার্চ এবং রিপ্লেস

সার্চ-রিপ্লেসকে ভিমের ভাষায় সাবস্টিটিউশন। একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ vim vimfoo.rc
```

আমাদের আগের সেভ করা ফাইলটিতে লেখা ছিল:

```
Best of luck for Tigers in ICC World Cup 2015.
```

আমরা যদি 'Tigers' কে 'TIGERS' দ্বারা প্রতিস্থাপিত করতে চাই তাহলে কমান্ড মোডে প্রথমে কনসোল ওপেন করবো ':' চেপে। তখন নীচে ':' চিহ্ন আসবে। তারপর লিখবো:

```
:%s/Tigers/TIGERS/g
```

এবং এন্টার চাপবো। এবার কমান্ডটির বিভিন্ন অংশ দেখা যাক:

অংশ	অর্থ
:	'.' চিহ্ন দিয়ে এক্স-কমান্ড কনসোল চালু করতে হয়।
%	এটা দিয়ে সাবস্টিটিউশন রেঞ্জ বোঝানো হয়। '%' চিহ্ন দিয়ে সম্পূর্ণ ফাইল বোঝানো হয়। 1,5 লিখলে প্রথমলাইন থেকে ৫ম লাইন পর্যন্ত বোঝাবে। আবার 10,\$ লিখলে ১০ম লাইন থেকে শেষ লাইন পর্যন্ত বোঝাত।
s	আমরা কী করতে চাইছি তা বলা। এখানে সাবস্টিটিউশন তাই s।
/Tigers/TIGERS/	লেখায় ৩টা '/' দিয়ে দুটো অংশকে আলাদা রাখা হয়েছে। প্রথম অংশে যেটা খুঁজতে হবে (এখানে Tigers) এবং শেষ অংশে তারবদলে যেটা বসাতে হবে (এখানে TIGERS)।
g	এই অংশটি অপশনাল। এটি লিখলে রেঞ্জের মধ্যে যতজায়গায় সার্চ-স্ট্রিংটি পাবে, রিপ্লেস করবে। এটি না দিলে শুধু প্রথমটি রিপ্লেস করত।

এন্টার চাপলে এরকম কনফার্মেশন চাইবে:

```
replace with Line (y/n/a/q/l/^E/^Y)?
```

'y' চাপলে কাজটি সম্পন্ন হবে। আমরা অন্য কী-গুলোর অর্থ জেনে নিই:

কী	অর্থ
y	সাবস্টিটিউশন করবে।
n	সাবস্টিটিউশন এর জন্য বর্তমানে নির্ধারিত প্যাটার্নটি স্কিপ করে পরেরটিতে যাবে।
a	সবগুলো সাবস্টিটিউশন সম্পন্ন করবে।
q	সাবস্টিটিউশন কনসোল বন্ধ করবে।
l	বর্তমান সাবস্টিটিউশনটি করে বন্ধ করবে।
Ctrl-e, Ctrl-y	নীচে ও উপরে স্ক্রল করবে।

## ভিম: একাধিক ফাইল নিয়ে কাজ করা

ভিমে একসাথে আপনি একাধিক ফাইল খুলতে চাইলে আপনার কমান্ড কাঠামোটি হবে এরকম:

```
vim file1 file2 file3...
```

অর্থাৎ, ভিমের আর্গুমেন্ট হিসেবে ফাইলগুলোর নাম। আসুন, ব্যবহারিক সুবিধার জন্য আমরা দুটো ফাইল তৈরী করি:

```
me@howtocode-pc:~$ ls -l /bin > ls-bin.txt
me@howtocode-pc:~$ ls -l /sbin > ls-sbin.txt
```

এবার ফাইলদুটি আমরা ভিম দিয়ে খুলবো:

```
me@howtocode-pc:~$ vim ls-bin.txt ls-sbin.txt
```

আমরা স্ক্রীনে প্রথম ফাইলটি অর্থাৎ ls-bin.txt দেখতে পারবো। আমরা যদি পরের ফাইলে যেতে চাই তাহলে তার জন্য কমান্ড হবে `:n` আর পূর্ববর্তী ফাইলে যেতে `:N`। উল্লেখ্য, আপনি ফাইল এডিট করার পর সেভ না দিলে ফাইল(বাফার) চাইলে আপনাকে বাধা দেবে। জোরপূর্বক যেতে সাথে `!` যোগ করতে হবে।

তাছাড়াও আপনি `:buffers` কমান্ড দিলে বাফার লিস্ট দেখাবে। এক্ষেত্রে বলে রাখি, ভিম কোনো ফাইল সরাসরি এডিট করে না বরং তার প্রতিলিপি ব্যবহার করে। এই প্রতিলিপিটিই বাফার। আমরা এডিট করার সময় আসলে বাফার এডিট করি। সেভ দিলে মূল ফাইলটি বাফার দ্বারা প্রতিস্থাপিত হয়। এখন `:buffers` কমান্ড দিলে এমনকিছু দেখবেন:

```
:buffers
 1 %a "ls-bin.txt" line 1
 2 "ls-sbin.txt" line 0
Press ENTER or type command to continue
```

এখন আপনি এন্টার চেপে বাফারলিস্ট বন্ধ করতে পারেন বা `:buffer` কমান্ডের সাথে নাম্বার ব্যবহার করে সেই বাফারে যেতে পারেন। যেমন ২য় ফাইল অর্থাৎ ls-sbin.txt এ যেতে গেলে আপনাকে লিখতে হবে `:buffer 2`

ভিম চালু থাকা অবস্থায় আপনি যদি আরেকটি ফাইল খুলতে চান তাহলে আপনাকে `:e` কমান্ড ব্যবহার করতে হবে। অর্থাৎ যদি আপনার ফাইলটির নাম হয় another.txt তাহলে কমান্ডটি হবে:

```
:e another.txt
```





# ইম্যাকস্ (Emacs)

ইম্যাকস্ আসলে নির্দিষ্ট একটি এডিটর নয় বরং একটি এডিটর পরিবার। এর মধ্যে সবচেয়ে জনপ্রিয় জিএনইউ ইম্যাকস্। আমাদের ব্যবহৃত পূর্ববর্তী এডিটরের তুলনায় ইম্যাকস্ একটু ভারি, রিসোর্স হাংরি। তবে আধুনিক কম্পিউটার (আপনার কম্পিউটারের র‍্যাম ৫১২ এমবি হলেও এক্ষেত্রে আধুনিক) এর ক্ষেত্রে এটি কোনো সমস্যা না।

ইম্যাকস্ তৈরী করা হয়েছে সি ও ইলিম্প ব্যবহার করে। সি দিয়ে এর মূল একটি অংশ তৈরী করার পর ইলিম্প (ইম্যাকস্ এর জন্য বিশেষ লিম্প। লিম্প একটি খুবই ডায়নামিক ইন্টারপ্রেটেড ল্যান্ডুয়েজ। এর গাণিতিক ঐতিহ্যের জন্য এটি সমাদৃত।) দিয়ে পুরো ইম্যাকস্ তৈরী করা হয়েছে। তাই খুব সহজে ইম্যাকস্ দিয়ে এমন অনেককিছু করিয়ে নেওয়া যায় যা অনেকসময় আমাদের ভাবনার বাইরে। এর আছে অসংখ্য প্লাগিন ও কার্যকারিতা। যার অধিকাংশই আপনার হয়ত কোনো কাজেই লাগবে না। বেসিকটি জানার পর আপনার প্রয়োজন অনুযায়ী অংশটুকু জেনে নেবেন।

এই কোর্সে আমরা জিএনইউ ইম্যাকস্ ২৪ ব্যবহার করছি। ইন্সটল করতে:

ডেবিয়ান সিস্টেমে: `# apt-get install emacs24` রেড হ্যাট সিস্টেমে: `# yum install emacs24` আর্চ লিনাক্স সিস্টেমে: `# pacman -S emacs24`

**\*\*বিঃদ্রঃ** আপনি যদি কমান্ডলাইন ছাড়া গ্রাফিক্যালি ইম্যাকস্ ব্যবহার না করতে চান তাহলে emacs24 এর জায়গায় emacs24-nox লিখতে পারেন। তাতে কিছু মেগাবাইট বেঁচে যাবে।

- **ইম্যাকস্: প্রথম ধাপ:** ইম্যাকস্ চালু, ইন্টারফেস পরিচিতি ও বন্ধ।
- **ইম্যাকস্: ক্যারেটর, কী এবং কমান্ড:** ক্যারেটর, কী এবং কমান্ড সম্পর্কিত আলোচনা।
- **ইম্যাকস্: বেসিক এডিটিং:** ইম্যাকসে বেসিক এডিটিং।
- **ইম্যাকস্: সার্চ এ্যান্ড রিপ্লেস:** ইম্যাকসে সার্চ এবং রিপ্লেস।
- **ইম্যাকস্: একাধিক ফাইল এডিট করা:** ইম্যাকসে একাধিক ফাইল নিয়ে কাজ করা।

# ইম্যাকস্: প্রথম ধাপ

## ইম্যাকস্ চালু করা

ইম্যাকস্ সাধারণভাবে চালু করলে তা গ্র্যাফিকালি চালু হবে। কমান্ড লাইনে চালু করতে আপনাকে লিখতে হবে:

```
emacs -nw |
```

## ইন্টারফেস পরিচিতি

চালু করার পর আপনি এমনকিছু দেখতে পাবেন:

```
File Edit Options Buffers Tools Help
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

Get help          C-h (Hold down CTRL and press h)
Emacs manual      C-h r      Browse manuals    C-h i
Emacs tutorial    C-h t      Undo changes     C-x u
Buy manuals      C-h RET    Exit Emacs       C-x C-c
Activate menubar M-`
(`C-' means use the CTRL key.  `M-' means use the Meta (or Alt) key.
If you have no Meta key, you may instead type ESC followed by the character.)
Useful tasks:
Visit New File          Open Home Directory
Customize Startup       Open \*scratch\* buffer

GNU Emacs 24.3.1 (x86_64-pc-linux-gnu, GTK+ Version 3.10.7)
  of 2014-03-07 on lamiak, modified by Debian
Copyright (C) 2013 Free Software Foundation, Inc.
-UUU:%%--F1 \*GNU Emacs\*   Top (1,0)   (Fundamental) -----
For information about GNU Emacs and the GNU system, type C-h C-a.
```

প্রথম লাইনে সাদা ব্যাকগ্রাউন্ডের ওপর কালো অক্ষরে লেখা থাকবে:

```
File Edit Options Buffers Tools Help
```

এটা হচ্ছে মেন্যুবার। আপনি **F10** চেপে মেন্যু ব্যবহার করতে পারেন। তারপর এই অংশটুকু আপনার এডিটর উইন্ডো:

```
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
```

```
Get help          C-h (Hold down CTRL and press h)
Emacs manual      C-h r      Browse manuals    C-h i
Emacs tutorial    C-h t      Undo changes    C-x u
Buy manuals       C-h RET     Exit Emacs      C-x C-c
Activate menubar  M-`
(`C-' means use the CTRL key. `M-' means use the Meta (or Alt) key.
If you have no Meta key, you may instead type ESC followed by the character.)
Useful tasks:
Visit New File          Open Home Directory
Customize Startup       Open \*scratch\* buffer

GNU Emacs 24.3.1 (x86_64-pc-linux-gnu, GTK+ Version 3.10.7)
of 2014-03-07 on lamiak, modified by Debian
Copyright (C) 2013 Free Software Foundation, Inc.
```

কোনো ফাইল না খুললে শুধু ইম্যাকস্ চালু করলে শুরুতে এই ইন্ট্রোডাকশন স্ক্রীন দেখাবে। এখানে কিছু গুরুত্বপূর্ণ শর্টকাট দেওয়া আছে। হয়ত আপনি ইতমধ্যে লক্ষ্য করেছেন `c-` মানে CTRL এবং `m-` মানে Meta বা Alt। একইভাবে RET হল Return বা Enter।

তারপরেই আপনি পাবেন মোড লাইন (Mode line):

```
-UUU:%%--F1 \*GNU Emacs\* Top (1,0) (Fundamental) -----
```

এর অংশগুলোর সংক্ষিপ্ত অর্থ এরকম:

- **-UUU:%%--F1**: ক্যারেটের এনকোডিং, লাইনব্রেক মেথড ইত্যাদি সম্পর্কিত তথ্য।
- **\*GNU Emacs\***: বাফারের নাম।
- **Top (1,0)**: কার্সরের অবস্থান। এখানে বলছে কার্সর ডকুমেন্টের উপরের দিকে আছে (Top)। তারপরেই লেখা আছে (1,0)। এই সংখ্যাদ্বয়ের প্রথমটি অর্থাৎ 1 লাইন নম্বর এবং দ্বিতীয়টি অর্থাৎ 0 দ্বারা কলাম নম্বর বোঝানো হয়েছে।
- **(Fundamental)**: এখানে ডকুমেন্টের মোড বলা হয়েছে। ইম্যাকস্ অসংখ্য ডকুমেন্ট মোড আছে। যা ফাইলের এক্সটেনশন অনুযায়ী বা ইউজারের ইচ্ছায় কার্যকর হয় এবং বিশেষ বিশেষ সুবিধা দেয়। যেমন পাইথন মোডে অটোমেটিক কোড ইনডেন্টেড হবে। চার স্পেস পরিমাণ ট্যাব দিয়ে আবার রুবি মোডে সেটি হবে দুই স্পেস।

সর্বশেষ লাইনটি হল ইকো এরিয়া:

```
For information about GNU Emacs and the GNU system, type C-h C-a.
```

এখানে প্রয়োজন মত আপনাকে বিভিন্ন জিনিস জানানো হবে। তাছাড়া `m-x` চাপলে এখানে কমান্ড নিতে মিনিবাফার চালু হবে।

## ইম্যাক্স বন্ধ করা

ইম্যাক্স বন্ধ করতে `C-x C-c` অর্থাৎ প্রথমে `C-x` চাপুন তারপর `C-c` ।

## ইম্যাক্স: ক্যারেটর, কী এবং কমান্ড

ইম্যাক্স ভালোভাবে ব্যবহার করতে হলে এবং কোর্সের সুবিধার্থে এখানে প্রাথমিক তিনটি বিষয় ব্যাখ্যা করা প্রয়োজন। আসুন, তবে দেখা যাক।

### ক্যারেটর (Character)

এককথায় কীবোর্ডের প্রতিটি বোতামে এক বা একাধিক ক্যারেটর আছে। এর আছে কিছু শ্রেণীবিভাগ:

ক্যারেটর এর ধরণ	উদাহরণ	ব্যাখ্যা
সাধারণ ক্যারেটর	অ, ক, A, c, =, (, ] SPC(স্পেস) ইত্যাদি।	সাধারণত আমরা লেখাজোকায় যেসব ক্যারেটর ব্যবহার করি তাদের সাধারণ ক্যারেটর বলা হয়।
বিশেষ বা স্পেশাল ক্যারেটর	TAB (ট্যাব), RET, DEL, ESC, F1 থেকে F12, Home, left ইত্যাদি।	সাধারণ ক্যারেটর ও মোডিফায়ার ছাড়া বাকি সব স্পেশাল ক্যারেটর। এগুলো বিভিন্ন বিশেষ কাজ করে থাকে।
মোডিফায়ার ক্যারেটর	CTRL এবং META বা ALT	CTRL এবং ALT (যাকে METAও বলা হয়।) এ দুটি মোডিফায়ার। এরা সাধারণ ও স্পেশাল ক্যারেটরগুলোকে বিশেষ কাজ করিয়ে নেয়। ইম্যাক্সে মোডিফায়ার এর প্রচুর ব্যবহার হয়ে থাকে।

### কী (Key)

সহজভাষায়, ইম্যাক্সে আমাদের ব্যবহারের জন্য যেসব শর্টকাট আছে সেগুলোই কী। আমরা কোর্সে বিভিন্ন জায়গায় এরকম অনেক কী'র সাথে পরিচিত হব। একটা কী'তে কয়েকটি অংশ থাকতে পারে। যেমন: `C-x C-f` এ দুটি অংশ। প্রথমে CTRL চেপে x তারপর CTRL চেপে f। একইভাবে M-x মানে META বা ALT চেপে f। আগেও বলেছি 'C-' ও 'M-' মানে যথাক্রমে CTRL এবং META। কী দুইরকম:

- **সম্পূর্ণ কী (Complete key):** যে কী কোনো কাজ করতে সক্ষম তাকে আমরা কম্প্লিট কী বলি। যেমন: `C-x C-f`। এটি দিয়ে আপনি এডিটরে নতুন ফাইল খুলতে পারবেন।
- **উপসর্গ কী (Prefix key):** আপনারা হয়ত বাঙলা ব্যকরণে উপসর্গ পড়েছেন। ব্যাপারটা একইরকম। এগুলো নিজে কিছু করতে পারে না। বরং কমান্ডের পরবর্তী অংশের জন্য অপেক্ষা করে। যেমন আপনি শুধু `C-x` চাপলে ইম্যাক্স কমান্ডের পরবর্তী অংশের জন্য অপেক্ষা করবে।

### কমান্ড (Command)

ইম্যাক্স এর কাছে কীগুলোর বিশেষ মানে নেই। আপনি চাইলেই সেটা চেঞ্জ করে অন্যকিছু করতে পারেন। আসলে গুরুত্বপূর্ণ হচ্ছে শর্টকাট এর পিছনের কমান্ডটি। এই কমান্ডগুলো ইলিম্পে লেখা একেকটি ফাংশন এবং `M-x` চেপে এদের নাম লিখে আপনি কমান্ডগুলি কার্যকর করতে পারেন। আমরা পরবর্তীতে প্রত্যেকটি কী-এর পাশে ব্রাকেটে

কমান্ডটিও লিখে দেবো।

# ইম্যাকস্: বেসিক এডিটিং

## ফাইল খোলা

ইম্যাকসে আপনি ন্যানো এবং ভিমের মতই কোনো ফাইল খুলতে পারেন:

```
emacs -nw file...
```

এভাবে আমরা emacsfoo.txt ফাইলটি খুলবো:

```
me@howtocode-pc:~$ emacs -nw emacsfoo.txt
```

অথবা আপনার emacs চালু থাকলে `c-x c-f` (find-file) চাপুন। দেখবে ইকো এরিয়ায় এরকম কিছু আসবে:

```
Find file: ~/
```

এরপর লিখুন emacsfoo.txt। অর্থাৎ ইকো এরিয়ায় এরকম দেখাবে:

```
Find file: ~/emacsfoo.txt
```

এন্টার চাপুন।

বলাবাহুল্য, ওই নামে ফাইল না থাকলে ফাইল তৈরী করে নেবে।

## লেখালেখি

লেখালেখি শেখার বিশেষ কিছু নেই। সাধারণ এডিটরের মতই লিখতে পারেন। এ্যারো কী চেপে লেখার মধ্যে কার্সর ঘোরাতে পারেন। ব্যাকস্পেস চেপে মুছতে পারেন।

## সেভ

সেভ দিতে চাপুন `c-x c-s` (save-buffer)

## মার্ক, কাট, কপি, পেস্ট

ভিমের মতই ইম্যাকসের বেশকিছু কমান্ড আছে কাট ও কপির জন্য। কোনোটা একটা অক্ষর বা লাইন বা শব্দ কাট বা কপি করে। আসলে এতোগুলো মনে রাখাও সমস্যার। আমরা অধিকাংশ লেখা সিলেক্ট করে বা মার্ক করে কাট বা কপি করতে অভ্যস্ত। সুতরাং আপনাকে যা করতে হবে তা হল:

- প্রথমে মার্ক করতে হবে প্রয়োজনীয় অংশটুকু। মার্ক করতে যেখান থেকে মার্ক করা শুরু করতে চান সেখানে কার্সর নিন এবং `C-SPC` (`set-mark-command`) চাপতে হবে। তারপর এ্যাবো কী দিয়ে কার্সর সরিয়ে দরকারি অংশটুকু মার্ক করতে হবে।
- তারপর কাট/ডিলিট করতে `C-W` (`kill-region`) বা কপি করতে `M-W` (`kill-ring-save`) চাপতে হবে।
- যেখানে পেস্ট করবেন সেখানে গিয়ে `C-Y` (`yank`) চাপলে পেস্ট হবে।

## CUA mode

আপনি চাইলে `C-X`, `C-C` ও `C-V` দিয়ে যথাক্রমে কাট, কপি ও পেস্ট করতে পারেন। এর জন্য আপনার হোমে থাকা `.emacs` ফাইলটি খুলুন এবং নীচের লাইনটি লিখে সেভ দিন:

```
(cua-mode 1)
```

## বন্ধ করা

ইম্যাকস বন্ধ করতে `C-X C-C` (`save-buffers-kill-terminal`) চাপুন। এডিটর জন্য খোলা ফাইলগুলো সেভ করা থাকলে ইম্যাকস বন্ধ হবে নতুবা মিনিবাফারে সেভ করার জন্য প্রয়োজনীয় নির্দেশ চাইবে।



# ইম্যাক্স: সার্চ এ্যান্ড রিপ্লেস

## সার্চ

ইম্যাক্সে সার্চের বহুল ব্যবহৃত কী দুটি হল `c-s` (isearch-forward) এবং `c-r` (isearch-backward)। এদুটি কীবোর্ড থেকে টাইপ করা আপনার লেখা পড়বে এবং পরবর্তী বা পূর্ববর্তী যেখানে সেই লেখা পাবে সেখানে কার্সর নিয়ে যাবে। মনে করুন আপনি `c-s` চেপে isearch-forward চালু করলেন। এবার লিখলেন 'F' তাহলে ডকুমেন্টে পরবর্তী যেখানে 'F' আছে, সেখানে কার্সর যাবে। আবার লিখলেন 'O' তাহলে যেখানে 'FO' আছে সেখানে যাবে। লেখায় ভুল হলে DEL চেপে একটি করে ক্যারেক্টার মুছতে পারেন। সার্চ করে দরকারি জিনিসটা পেলে এন্টার চাপলে সার্চ বন্ধ হবে।

কিন্তু একই ডকুমেন্টে আপনার সার্চ স্ক্রিংয়ের সাথে একাধিক লেখা মিলতে পারে। পরেগুলো খুঁজতে হলে সার্চ কমান্ডটির পুনরাবৃত্তি করলেই হবে। এবার আর নতুন করে লিখতে হবে না। মনে করুন আপনি লেখার মধ্যে 'FOO' খুঁজছেন `c-s` দিয়ে। পরেরটিতে যেতে শুধু `c-s` চাপলেই হবে।

## সার্চ এবং রিপ্লেস

সার্চ এবং রিপ্লেসের জন্য সবচেয়ে ভালো কমান্ড হলো `m-%` (query-search)। `m-%` চাপলে প্রথমে আপনার কাছে কী সার্চ করতে চান জানতে চাইবে। আপনি মিনিবাফারে সেটি লিখে এন্টার দিলে কী দিয়ে রিপ্লেস করতে চান সেটি জানতে চাইবে। এটি লিখে এন্টার দিলে প্রত্যেকটি রিপ্লেসমেন্ট হাইলাইট করবে এবং আপনার কাছে অনুমতি চাইবে রিপ্লেস করবে কিনা। আপনি 'y' চাপলে রিপ্লেস করবে ও পরবর্তী সার্চ ম্যাচের কাছে যাবে। একইভাবে 'n' চাপলে রিপ্লেস না করে পরবর্তী টার্গেটে যাবে। সব রিপ্লেস করতে চাইল চাপবেন '!' আর বন্ধ করতে RET বা ESC।

# ইম্যাকস্: একাধিক ফাইল এডিট করা

শুরুতেই আমরা দুটো ধারণার সাথে পরিচিত হই। বাফার এবং উইন্ডো।

## বাফার (Buffer)

বাফার সম্পর্কে আমরা ভিম-এও কথা বলেছি। ভিম, ইম্যাকস্ এই এডিটরগুলো মূল ফাইলের একটি কপি নিয়ে কাজ করে এবং সেড দিলে কপিটি দ্বারা মূল ফাইলটি প্রতিস্থাপিত হয়। এই অস্থায়ী কপিটাই বাফার। আপনি একসাথে ইম্যাকসে একাধিক ফাইল খুলতে পারেন। তারা আলাদা আলাদা বাফার হিসেবে খুলবে। আপনি স্ক্রীনে মোড লাইনে আপনার বাফারের নাম দেখতে পারবেন।

## উইন্ডো (Window)

আপনি চাইলে আপনার স্ক্রীনকে উপর-নীচে পাশাপাশি একাধিক ভাগে ভাগ করে নিতে পারেন কাজের সুবিধার্থে। এই ভাগগুলোকে উইন্ডো বলে।

## একাধিক ফাইল খোলা

আপনি ইম্যাকসে একাধিক ফাইল চালু করার সময়ই খুলতে পারেন এভাবে:

```
emacs -nw files...
```

আবার চালু করা ইম্যাকস্ এ `C-x C-f` (find=file) চেপে এলটি একটি করে একাধিক ফাইল খুলতে পারেন।

## বাফার নিয়ন্ত্রণ

বাফার নিয়ন্ত্রণের জন্য বেশ কিছু কমান্ড আছে। তারমধ্যে সবচেয়ে জরুরিগুলো হল:

কী	কমান্ড	কার্যকারিতা
C-x LEFT	previous buffer	পূর্ববর্তী বাফারে যাবে।
C-x RIGHT	next buffer	পরবর্তী বাফারে যাবে।
C-x C- b	list- buffers	বাফারের একটি লিস্ট দেখাবে যেখান থেকে আপনি UP ও DOWN চেপে ও। এ্যারো কী চেপে বাফার সিলেক্ট করতে পারবেন।
C-x k	kill- buffer	আপনার কাছে জানতে চাইবে কোন বাফারটি বন্ধ করতে চান। কোনো বাফারের নাম না দিয়ে এন্টার চাপলে বর্তমান বাফার বন্ধ করবে।

## উইন্ডো নিয়ন্ত্রণ

প্রথমেই দেখে নিই কীভাবে স্প্লিট করে একাধিক উইন্ডো তৈরী করতে হয়:

কী	কমান্ড	কার্যকারিতা
C-x 2	split-window-below	উপর-নীচে দুটি উইন্ডো তৈরী করবে।
C-x 3	split-window-right	পাশাপাশি দুটি উইন্ডো তৈরী করবে।

নতুন তৈরী করা উইন্ডোগুলো একই ফাইল দেখাবে।

উইন্ডোগুলোর মধ্যে আপনি `c-x o` (other-window) চেপে সিলেক্ট করতে পারেন। ফাংশনটির নাম other-window হলেও একাধিক উইন্ডো থাকলে চক্রাকারে সেগুলি সিলেক্ট করবে।

উইন্ডো বন্ধ করতে আপনার দরকার হবে দুটি কী:

কী	কমান্ড	কার্যকারিতা
C-x 0	delete-window	সিলেক্টেড উইন্ডোটি বন্ধ করবে।
C-x 1	delete-other-windows	সিলেক্টেড উইন্ডো বাদে বাকি সব উইন্ডো বন্ধ করবে।

## অধ্যায় - তিন

# স্টোরেজ মিডিয়া

এই অধ্যায়ে আমরা স্টোরেজ মিডিয়া নিয়ে বিভিন্ন প্রাথমিক বিষয়গুলি জানবো। জানবো কীভাবে তাদের মাউন্ট ও আনমাউন্ট করতে হয়। এররের জন্য চেক করতে হয়। পার্টিশনিং এবং বিভিন্ন স্টোরেজ মিডিয়ার ব্যবহার।

- লিনাক্সের সাথে স্টোরেজ ডিভাইস: মাউন্ট ও আনমাউন্ট সম্পর্কিত প্রাথমিক ধারণা।
- মাউন্ট এবং আনমাউন্ট: মাউন্ট ও আনমাউন্ট করা।
- পার্টিশন এবং ফরম্যাট করা: পার্টিশন ও ফরম্যাট: fdisk ও mkfs এর ব্যবহার।
- ফাইলসিস্টেম টেস্ট এবং রিপেয়ার করা: fsck এর ব্যবহার।
- ডিভাইস ক্লোনিং: dd কমান্ডের ব্যবহার।
- ইমেজ তৈরী: সিডি/ডিভিডি ইমেজ তৈরী।
- অপটিক্যাল মিডিয়ায় রাইট করা: সিডি/ডিভিডি রাইট করা।

## লিনাক্সের চোখে স্টোরেজ ডিভাইস

আপনি যদি `ls -l /dev/` কমান্ডটি দেন তাহলে দীর্ঘ ডিভাইসের লিস্ট দেখতে পাবেন। এখানে পাবেন সকলরকম ডিভাইস। ডিভাইসগুলো কম্পিউটার বিভিন্নভাবে ব্যবহার করে। আমাদের বুঝতে নিশ্চয়ই অসুবিধা হয় না যে স্টোরেজ ডিভাইস কম্পিউটার কী কাজে ব্যবহার করে। নিশ্চয়ই তথ্য সংরক্ষণ ও তার থেকে পড়ার জন্য। লিনাক্সে কোনো স্টোরেজ ডিভাইস যুক্ত হলে সে কয়েকটি তথ্য সেটি সম্পর্কে জানতে পারে। ডিভাইসটির নাম, লেবেল, সেটির ফরম্যাট, uuid ইত্যাদি। এগুলো ব্যবহার করে লিনাক্স ডিভাইসটি নিয়ন্ত্রণ করতে পারে।

একটা জিনিস মাথায় রাখা দরকার যে একটা ডিভাইসে একাধিক পার্টিশন থাকলে সেগুলোকেও ডিভাইস হিসেবে দেখায়। আমাদের কাজের সময় আমরা সাধারণত ডিভাইসকে নাম দিয়ে প্রকাশ করি। এই নামটা সিস্টেম দেয়। এই নামকরণের পদ্ধতি খুব সহজ। যেমন আপনার প্রধান হার্ডডিস্ক বা স্টোরেজ ডিভাইস হল sda। এর প্রথম পার্টিশন যেটা উইন্ডোজে হয় সি ড্রাইভ সেটা হবে sda1। এভাবে পরপর পার্টিশনের নাম হবে। এরপর দ্বিতীয় ডিভাইস যুক্ত করলে, তা হতে পারে আরেকটা হার্ডডিস্ক বা পেনড্রাইভ, সেটি হবে sdb এবং তার পার্টিশনগুলো sdb1, sdb2... এভাবে। আবার সিডি/ডিভিডি অর্থাৎ অপটিক্যাল ড্রাইভগুলো sr0, sr1 এভাবে নাম দেওয়া হয়।

এখন প্রশ্ন হচ্ছে, ডিভাইসের তথ্য নিয়ন্ত্রণের জন্য লিনাক্সে কী করতে হয়? বা কী করলে আসলে আমরা ওই তথ্য ব্যবহার করতে পারি? আমরা ইত্যমধ্যে জেনেছি যে লিনাক্সের ফাইলসিস্টেম শুরু হয় রুট থেকে এবং ক্রমে শাখা-প্রশাখা বিস্তার করে। আমরা স্টোরেজ ডিভাইসের তথ্য নিয়ন্ত্রণ করতে চাইলে তাকে এই ফাইলসিস্টেমের কোথাও জুড়ে দিতে হবে। ঘাবড়াবার কিছু নেই। রোজ রোজ আপনাকে এটি করতে হয় না। আপনি পেনড্রাইভ বা সিডি ঢোকালে লিনাক্স নিজেই এটা করে নেয়। ফাইলসিস্টেমে স্টোরেজ ডিভাইস জুড়ে দেয়ার এই প্রক্রিয়াকে মাউন্ট (mount) করা বলে। এটা কীভাবে কাজ করে তা দেখতে আমরা /etc/fstab ফাইলটিতে চোখ বোলাবো। এটির কাজ হচ্ছে কম্পিউটার চালু হওয়ার সময় কার্নেলকে জানিয়ে দেওয়া কোন কোন ডিভাইস মাউন্ট করতে হবে। বলাবাহুল্য, আমাদের fstab ফাইল সবার একরকম হবে না।

```
me@howtocode-pc:~$ less /etc/fstab
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda6 during installation
UUID=5eef4c8c-6f5b-408b-bae6-9de6162611af / ext4 errors=remount-ro 0
# /home was on /dev/sda2 during installation
UUID=e42a0385-8d90-473b-907c-ce458c821739 /home ext4 defaults 0
# swap was on /dev/sda3 during installation
UUID=6fa0c755-8325-4740-9cc9-5e3de831b65d none swap sw 0
/etc/fstab (END)
```

ফাইলটির যেসব লাইন # দিয়ে শুরু হয়েছে সেগুলি কমেন্ট। অর্থাৎ কম্পিউটার ওগুলো আমলে আনে না। ইউজারের জন্য বিভিন্ন তথ্য দেওয়া থাকে। আমরা এই অংশটুকু বিবেচনায় আনি:

```
# <file system> <mount point> <type> <options> <dump> <pass>
# / was on /dev/sda6 during installation
UUID=5eef4c8c-6f5b-408b-bae6-9de6162611af / ext4 errors=remount-ro 0
```

প্রথম লাইনে fstab টেবিলের কলামগুলোর নাম দেওয়া আছে যেন আমরা সহজে বুঝতে পারি। তারপর আরেকটি কমেন্ট। যেখানে বলা আছে পরবর্তী লাইনে কী করা হচ্ছে। তারপর মূল এন্ট্রিটি। এবার এগুলোর অর্থ জানা যাক:

কলাম	মান	অর্থ
file system	UUID=5eef4c8c-6f5b-408b-bae6-9de6162611af	ডিভাইসের নাম বা uuid দেওয়া থাকে।
mount point	/	ফাইলসিস্টেমের কোনখানে মাউন্ট করবে অর্থাৎ মাউন্ট পয়েন্ট। এখানে ছিল '/' অর্থাৎ এই ড্রাইভটা রুটে মাউন্ট হবে।
type	ext4	পার্টিশনের ফরম্যাট, যেমন এটি ext4 পার্টিশন।
options	errors=remount-ro	মাউন্টের বিভিন্ন অপশন, থাকতেও পারে, নাও পারে।
dump	0	ডাম্প একটি ব্যাকআপ ইউটিলিটি। এটি চালু না বন্ধ থাকবে তা বলা থাকে। 1 হলে চালু, 0 হলে বন্ধ।
pass	1	fsck দিয়ে ফাইল চেকের নির্দেশনা। 0 হলে চেক করবে না। 1 দিলে সবার প্রথমে এটি চেক করবে। 2 দেওয়া পার্টিশনগুলো 1 এর পর চেক করবে।

একইভাবে দেখেছি sda2 পার্টিশনটি /home/ এ মাউন্ট করা হয়েছে। অর্থাৎ /home/ এবং এর ভিতরে থাকা ফোল্ডার ও ফাইলগুলো sda2 পার্টিশনে আছে।

একইভাবে। ফাইল সিস্টেম থেকে কোনো ডিভাইস বিযুক্ত করাকে বলে আনমাউন্ট (unmount) করা।

ডিভাইস আনমাউন্ট করা হলেও তাতে কিন্তু পাওয়ার সাপ্লাই থাকে। এপর্যায়ে ডিভাইসের ডাটাতে এক্সেস থাকে না। ফলে তার ম্যানিপুলেশনের সুবিধা তৈরী হয়। একটি ডিভাইসের সকল পার্টিশন আনমাউন্টেড হলে তবেই সেটার পার্টিশন টেবিল নিয়ে নাড়াচাড়া করা যায়।

আনমাউন্ট করা ডিভাইসে পাওয়ার সাপ্লাই বন্ধ করা তা সে ডিভাইসটি খুলে নেন বা ইজেক্ট কমান্ড দিয়েই করেন, তাকে ইজেক্ট করা বলে।

# মাউন্ট এবং আনমাউন্ট

লিনাক্স সিস্টেমে আমরা মাউন্ট এবং আনমাউন্ট করতে যথাক্রমে `mount` ও `umount` কমান্ড ব্যবহার করে থাকি।

আনমাউন্ট করতে হলে আপনাকে ডিভাইস বা পার্টিশনের নাম জানতে হবে। কোনো আগুর্মেণ্ট ছাড়া যদি `mount` কমান্ড দেন তাহলে এখন মাউন্টেড ডিভাইসগুলো দেখতে পাবেন:

```
me@howtocode-pc:~$ mount
/dev/sda6 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
none on /run/lock type tmpfs (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
none on /run/user type tmpfs (rw,noexec,nosuid,nodev,size=104857600,mode=0755)
none on /sys/fs/pstore type pstore (rw)
/dev/sda2 on /home type ext4 (rw)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,noexec,nosuid,nodev)
systemd on /sys/fs/cgroup/systemd type cgroup (rw,noexec,nosuid,nodev,none,name=systemd)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,user=nishadsingha)
/dev/sr1 on /media/me/Teletalk Modem type iso9660 (ro,nosuid,nodev,uid=1000,gid=1000,ioc
/dev/sdc1 on /media/me/Roy type fuseblk (rw,nosuid,nodev,allow_other,default_permissions,
```

আমরা আউটপুটের একদম শেষ লাইনে দেখতে পাচ্ছি যে আমার পেনড্রাইভ যেটার লেবেল হল **Roy** সেটাকে **/media/me/Roy** ফোল্ডারে মাউন্ট করা হয়েছে। এখান থেকে আমরা পেনড্রাইভের পার্টিশনের নামও পেয়ে গেছি। সেটি হল **/dev/sdc1**। বুঝতে অসুবিধা হয় না যে ডিভাইসটির নাম হবে তাহলে **/dev/sdc**। এখন আমরা যদি **sdc1** পার্টিশনটিকে আনমাউন্ট করতে চাই তাহলে লিখবো:

```
# umount/dev/sdc1
```

সামনে হ্যাশ(#) চিহ্ন দেখে আপনি নিশ্চয়ই বুঝতে পারছেন যে এটি আপনাকে সুপারইউজার হিসেবে দিতে হবে।

এবার মনে করুন আমরা এই ড্রাইভটাকে অন্য কোথাও মাউন্ট করতে চাই। `mount` কমান্ডের কমান্ড কাঠামো এরকম:

```
mount source_device mount_point
```

এটাকে তাহলে আমাদের হোম ফোল্ডারে `mounted_here` নামে ফোল্ডারে মাউন্ট করা যাক:

```
$ mkdir mounted_here  
# sudo mount /dev/sdc1 mounted_here
```

CD/DVD বা iso image এর জন্য মাউন্টের ধরণটা একটু আলাদা। মনে করি আপনার হোম ফোল্ডারে একটি আইএসও ইমেজ আছে। এটি আপনি মাউন্ট করতে চাইলে ইমেজের মধ্যকার ফাইলসিস্টেম টাইপ আপনাকে বলতে হবে। অধিকাংশ ইমেজ/CD/DVD iso9660 টাইপের হয়।

```
$ mkdir iso_mount  
# mount -t iso9660 -o loop ~/ubuntu-gnome-14.04-desktop-amd64.iso iso_mount/  
mount: block device ~/ubuntu-gnome-14.04-desktop-amd64.iso is write-protected, mounting r
```

প্রথমে আমরা `iso_mount` বলে একটা ফোল্ডার তৈরী করেছি। এবার আমরা মাউন্ট কমান্ডের সাথে `-t` অপশন দিয়ে টাইপটি লিখেছি। `-o` এর মাধ্যমে আমরা `loop` অপশন যোগ করেছি। যার ফলে এটিকে একটি ডিভাইস হিসেবে কম্পিউটার ধরে নেবে। এই অপশনটি না দিলেও হয় তবে কখনো কখনো কাজে লাগে। তারপর আইসও ফাইলের নাম ও মাউন্টপয়েন্ট ফোল্ডারটির নাম দিয়েছি।



## পার্টিশন এবং ফরম্যাট করা

অতিরিক্ত সতর্কতা অবলম্বন করুন। ডিভাইসের নাম আমার ও আপনার ভিন্ন হতেই পারে। এটি বিবেচনায় রাখবেন নতুবা বড়সড় ভাটা লস হতে পারে।

এই লেসনের জন্য আমি ১৬গিগাবাইটের একটি পেনড্রাইভ ব্যবহার করছি। আমরা এখন একটি ৪ জিবি ext4 এবং একটি ১২জিবি NTFS পার্টিশন তৈরী করবো। পেনড্রাইভটি চুকিয়ে প্রথমে `mount` কমান্ড দিয়ে দেখে নেবো ডিভাইসের নামটি:

```
$ mount
/dev/sda6 on / type ext4 (rw,errors=remount-ro)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
none on /sys/fs/cgroup type tmpfs (rw)
none on /sys/fs/fuse/connections type fusectl (rw)
none on /sys/kernel/debug type debugfs (rw)
none on /sys/kernel/security type securityfs (rw)
udev on /dev type devtmpfs (rw,mode=0755)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=0620)
tmpfs on /run type tmpfs (rw,noexec,nosuid,size=10%,mode=0755)
none on /run/lock type tmpfs (rw,noexec,nosuid,nodev,size=5242880)
none on /run/shm type tmpfs (rw,nosuid,nodev)
none on /run/user type tmpfs (rw,noexec,nosuid,nodev,size=104857600,mode=0755)
none on /sys/fs/pstore type pstore (rw)
/dev/sda2 on /home type ext4 (rw)
binfmt_misc on /proc/sys/fs/binfmt_misc type binfmt_misc (rw,noexec,nosuid,nodev)
systemd on /sys/fs/cgroup/systemd type cgroup (rw,noexec,nosuid,nodev,none,name=systemd)
gvfsd-fuse on /run/user/1000/gvfs type fuse.gvfsd-fuse (rw,nosuid,nodev,user=nishadsingha)
/dev/sr1 on /media/me/Teletalk Modem type iso9660 (ro,nosuid,nodev,uid=1000,gid=1000,ioc
/dev/sdc1 on /media/me/Roy type fuseblk (rw,nosuid,nodev,allow_other,default_permissions,
```

সবচেয়ে নীচের লাইনে আমরা আমাদের পেনড্রাইভটির একমাত্র পার্টিশনটি দেখতে পাচ্ছি যার নাম sdc1। তারমানে আমাদের পুরো ডিভাইসটি sdc। এখন sdc এর পার্টিশন টেবিল নতুন করে করতে হলে আমাদের আগে মাউন্ট থাকা পার্টিশনগুলো আনমাউন্ট করতে হবে এভাবে:

```
# umount /dev/sdc1
```

## পার্টিশন তৈরী করা

আমরা পার্টিশন করতে `fdisk` প্রোগ্রামটি ব্যবহার করবো। এজন্য আমাদের ডিভাইসের নামটি আর্গুমেন্ট হিসেবে দিতে হবে:

```
# fdisk /dev/sdc
```

ফলে এরকম একটা লাইন আসবে:

```
Command (m for help):
```

**m** চেপে এন্টার দিয়ে আমরা কমান্ডগুলি দেখতে পারি:

```
Command (m for help): m
Command action
  a   toggle a bootable flag
  b   edit bsd disklabel
  c   toggle the dos compatibility flag
  d   delete a partition
  l   list known partition types
  m   print this menu
  n   add a new partition
  o   create a new empty DOS partition table
  p   print the partition table
  q   quit without saving changes
  s   create a new empty Sun disklabel
  t   change a partition's system id
  u   change display/entry units
  v   verify the partition table
  w   write table to disk and exit
  x   extra functionality (experts only)
```

```
Command (m for help):
```

আমরা দেখতে পাচ্ছি **p** চাপলে বর্তমান পার্টিশন টেবিল প্রিন্ট করবে বা দেখাবে। এটা দেখে নিই:

```
Command (m for help): p

Disk /dev/sdc: 16.1 GB, 16125001728 bytes
255 heads, 63 sectors/track, 1960 cylinders, total 31494144 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x87a76b87
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1		2048	31494143	15746048	7	HPFS/NTFS/exFAT

```
Command (m for help):
```

প্রথম কয়েকলাইনে ডিভাইসসম্পর্কিত বিভিন্ন তথ্যের পর একটা লাইন ছেড়ে দিয়ে পার্টিশন টেবিল দেখাচ্ছে। পার্টিশন টেবিলে আমরা এখন একটাই পার্টিশন দেখছি sdc1।

এবার আমরা **o** চেপে একটা পার্টিশন টেবিল তৈরী করবো:

```
Command (m for help): o
Building a new DOS disklabel with disk identifier 0xe981134c.
Changes will remain in memory only, until you decide to write them.
After that, of course, the previous content won't be recoverable.

Warning: invalid flag 0x0000 of partition table 4 will be corrected by w(rite)

Command (m for help):
```

এবার **p** দিলে আমরা ফাঁকা পার্টিশন টেবিল পাবো এরকম:

```
Command (m for help): p

Disk /dev/sdc: 16.1 GB, 16125001728 bytes
255 heads, 63 sectors/track, 1960 cylinders, total 31494144 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xe981134c
```

Device	Boot	Start	End	Blocks	Id	System

```
Command (m for help):
```

এবার আমরা ৪জিবির ext4 পার্টিশন তৈরী করবো। তারজন্য **n** কমান্ড দিতে হবে:

```
Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-31494143, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-31494143, default 31494143): +4G
```

প্রথমে **n** দেওয়ার পর জানতে চেয়েছে আমরা প্রাইমারি না এক্সটেন্ডেড পার্টিশন করতে চাই। আমরা **p** দিয়ে প্রাইমারি সিলেক্ট করেছি। তারপর পার্টিশন নাম্বার জানতে চেয়েছে এক থেকে চারের মধ্যে কেননা ৪টির বেশি প্রাইমারি পার্টিশন করা যায় না। আমরা এন্টার চাপলে ডিফল্টভাবে 1 নিত তবুও আমরা 1 দিয়েছি। তারপর পার্টিশনের শুরুটা জানতে চেয়েছে। আমরা এন্টার চেপে ডিফল্টটা সিলেক্ট করেছি। তারপর আমাদের কাছে লাস্ট সেক্টর জিজ্ঞাসা করা হয়েছে। আমরা সেক্টর শুধু সংখ্যায় প্রকাশ করতে পারতাম। বা সংখ্যার সাথে K, M বা G দিয়ে যথাক্রমে কিলোবাইট, মেগাবাইট ও গিগাবাইট বলতে পারি। আমরা +4G দিয়ে বলেছি শুরু থেকে ৪ জিবি পর্যন্ত এই পার্টিশনের সীমানা। একইভাবে আমরা বাকি অংশ থেকে দ্বিতীয় পার্টিশন বানাবো:

```

Command (m for help): n
Partition type:
   p   primary (1 primary, 0 extended, 3 free)
   e   extended
Select (default p): p
Partition number (1-4, default 2): 2
First sector (8390656-31494143, default 8390656):
Using default value 8390656
Last sector, +sectors or +size{K,M,G} (8390656-31494143, default 31494143):
Using default value 31494143
Command (m for help):

```

এবার আমরা পার্টিশনের শুরু ও শেষ উভয়ক্ষেত্রে ডিফল্ট মান ব্যবহার করেছি যার ফলে বাকি পুরো ফাঁকা জায়গা ব্যবহার করে পার্টিশন তৈরী করেছে।

এবার আমরা পার্টিশন টেবিলটি দেখি:

```

Command (m for help): p

Disk /dev/sdc: 16.1 GB, 16125001728 bytes
255 heads, 63 sectors/track, 1960 cylinders, total 31494144 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xe981134c

   Device Boot      Start         End      Blocks   Id  System
/dev/sdc1             2048        8390655        4194304    83  Linux
/dev/sdc2        8390656       31494143       11551744    83  Linux

Command (m for help):

```

আমরা দেখতে পাচ্ছি দুটোরই System 'Linux'। কিন্তু দ্বিতীয়টি আমরা NTFS করতে চাই তাই তার id পরিবর্তন করতে হবে। এরজন্য আমরা `l` কমান্ড দিয়ে জানা পার্টিশন টাইপের লিস্টটি দেখবো:

```
Command (m for help): l
```

0 Empty	24 NEC DOS	81 Minix / old Lin	bf Solaris
1 FAT12	27 Hidden NTFS Win	82 Linux swap / So	c1 DRDOS/sec (FAT-
2 XENIX root	39 Plan 9	83 Linux	c4 DRDOS/sec (FAT-
3 XENIX usr	3c PartitionMagic	84 OS/2 hidden C:	c6 DRDOS/sec (FAT-
4 FAT16 <32M	40 Venix 80286	85 Linux extended	c7 Syrix
5 Extended	41 PPC PReP Boot	86 NTFS volume set	da Non-FS data
6 FAT16	42 SFS	87 NTFS volume set	db CP/M / CTOS / .
7 HPFS/NTFS/exFAT	4d QNX4.x	88 Linux plaintext	de Dell Utility
8 AIX	4e QNX4.x 2nd part	8e Linux LVM	df BootIt
9 AIX bootable	4f QNX4.x 3rd part	93 Amoeba	e1 DOS access
a OS/2 Boot Manag	50 OnTrack DM	94 Amoeba BBT	e3 DOS R/O
b W95 FAT32	51 OnTrack DM6 Aux	9f BSD/OS	e4 SpeedStor
c W95 FAT32 (LBA)	52 CP/M	a0 IBM Thinkpad hi	eb BeOS fs
e W95 FAT16 (LBA)	53 OnTrack DM6 Aux	a5 FreeBSD	ee GPT
f W95 Ext'd (LBA)	54 OnTrackDM6	a6 OpenBSD	ef EFI (FAT-12/16/
10 OPUS	55 EZ-Drive	a7 NeXTSTEP	f0 Linux/PA-RISC b
11 Hidden FAT12	56 Golden Bow	a8 Darwin UFS	f1 SpeedStor
12 Compaq diagnost	5c Priam Edisk	a9 NetBSD	f4 SpeedStor
14 Hidden FAT16 <3	61 SpeedStor	ab Darwin boot	f2 DOS secondary
16 Hidden FAT16	63 GNU HURD or Sys	af HFS / HFS+	fb VMware VMFS
17 Hidden HPFS/NTF	64 Novell Netware	b7 BSDI fs	fc VMware VMKCORE
18 AST SmartSleep	65 Novell Netware	b8 BSDI swap	fd Linux raid auto
1b Hidden W95 FAT3	70 DiskSecure Mult	bb Boot Wizard hid	fe LANstep
1c Hidden W95 FAT3	75 PC/IX	be Solaris boot	ff BBT
1e Hidden W95 FAT1	80 Old Minix		

```
Command (m for help):
```

লিস্টের 7 নম্বরটি অর্থাৎ HPFS/NTFS/exFAT এ আমাদের পরিবর্তন করতে হবে। এজন্য আমরা **t** চাপবো:

```
Command (m for help): t
Partition number (1-4): 2
Hex code (type L to list codes): 7
Changed system type of partition 2 to 7 (HPFS/NTFS/exFAT)
```

```
Command (m for help):
```

**t** দেওয়ার পর আমাদের পার্টিশনের নাম্বার জিজ্ঞাসা করেছে। আমরা দ্বিতীয়টি পরিবর্তন করবো তাই 2 দিয়েছি। এবার আমাদের কাছে পার্টিশনের ধরনের কোড জানতে চাইছে। আমরা পরিকল্পনামাফিক 7 দিয়েছি। এবার পার্টিশন টেবিল দেখি:

```
Command (m for help): p
```

```
Disk /dev/sdc: 16.1 GB, 16125001728 bytes
255 heads, 63 sectors/track, 1960 cylinders, total 31494144 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xe981134c
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1		2048	8390655	4194304	83	Linux
/dev/sdc2		8390656	31494143	11551744	7	HPFS/NTFS/exFAT

```
Command (m for help):
```

সব ঠিক আছে। এবার পরিবর্তন সংরক্ষণের জন্য **w** চেপে সেভ করি:

```
Command (m for help): w
```

```
The partition table has been altered!
```

```
Calling ioctl() to re-read partition table.
Syncing disks.
```

## ফাইলসিস্টেম তৈরী করা

এবার আমরা তৈরী করা পার্টিশন দুটি অর্থাৎ sdc1 ও sdc2 তে যথাক্রমে ext4 ও NTFS ফাইলসিস্টেম বানাবো(ফরম্যাট বলা যায় সহজভাষায়।) এজন্য আমরা `mkfs` কমান্ডটি ব্যবহার করবো। এর কাঠামোটি এরকম:

```
# mkfs -t partition_type Device
```

তাহলে, আমরা sdc1 এ ext4 করবো এভাবে:

```
# mkfs -t ext4 /dev/sdc1
mke2fs 1.42.9 (4-Feb-2014)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
262144 inodes, 1048576 blocks
52428 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=1073741824
32 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

একইভাবে আমরা sdc2কে NTFS করবো এভাবে:

```
# mkfs -t ntfs /dev/sdc2
```

এবার আমরা পেনড্রাইভটি কম্পিউটার থেকে খুলে আবার লাগালে আমাদের বর্তমান পার্টিশনগুলো দেখতে পাবো।

# ফাইলসিস্টেম টেস্ট এবং রিপেয়ার করা

## fsck

ফাইলসিস্টেমে স্বয়ংক্রিয়ভাবে সমস্যা নির্ণয় এবং তার সমাধান করার জন্য লিনাক্সে `fsck` ব্যবহার করা হয়। হার্ডওয়্যার এর ত্রুটিতে বিভিন্নরকম ডাটা করাপশনে এটি বেশ কাজে লাগে। উদ্ধারকৃত ফাইলসমূহ ওই ড্রাইভের `lost+found` ফোল্ডারে পাওয়া যায়।

যদি আমরা `sdb1` ড্রাইভে `fsck` ব্যবহার করতে চাই তাহলে আমাদের কমান্ড হবে:

```
# umount /dev/sdb1  
# fsck /dev/sdb1
```

তাছাড়াও সিস্টেমের ড্রাইভগুলো যেগুলো আপনি সিস্টেম চালু অবস্থায় আনমাউন্ট করে টেস্ট করতে পারছেন না সেগুলোকে কম্পিউটার চালুর সময় আমরা চেক করতে পারি রুট ডিরেক্টরিতে `forcefsck` নামে একটা ফাঁকা ফাইল তৈরী করে। সেক্ষেত্রে পরবর্তীবার কম্পিউটার চালু হওয়ার সময় `fsck` টেস্ট করবে।

```
# touch /forcefsck
```

## চেকসাম (checksum)

চেকসাম একধরনের ডাটা ইন্টিগ্রিটি পরীক্ষা করার উপায়। কোনো ফাইলের উপর বিশেষ গাণিতিক বিশ্লেষণ চালিয়ে চেকসামের একটি সংখ্যামান পাওয়া যায়। সেই তথ্যের একটি বিটও যদি এদিক-ওদিক হয় তবে চেকসাম মিলবে না। বিভিন্ন জায়গা থেকে ডাউনলোড করার সময় আপনি চেকসাম তথ্য পাবেন। যা দিয়ে আপনি ডাউনলোড করা ফাইলটি ঠিক আছে কিনা দেখতে পারেন। মনে করেন আপনি `image.iso` নামে একটি ফাইল ডাউনলোড করেছেন এবং তার `md5sum` (চেকসামের একটি ধরণ) সংখ্যাটিও পেয়েছেন। এবার আপনি নীচের কমান্ডটি দেবেন:

```
me@howtocode-pc:~$ md5sum image.iso
```

এবার আপনি `image.iso` এর চেকসাম সংখ্যা পাবেন। আপনি তখন দুটি সংখ্যা মিলিয়ে দেখতে পারবেন ফাইলটি ঠিক আছে কিনা।

চেকসামের আরেকটা গুরুত্বপূর্ণ ব্যবহার হল নতুন রাইট করা সিডি/ডিভিডির ডাটা ইন্টিগ্রিটি পরীক্ষা করা। তবে এক্ষেত্রে একটা জিনিস মনে রাখতে হবে যে আমরা সিডি বা ডিভিডির ততটুকুই ধর্তব্যে আনবো যতটা রাইট করা হয়েছে, পুরো সিডি/ডিভিডিটা নয়। অধিকাংশক্ষেত্রে যেমন ডিস্ক-এ্যাট-ওয়ান্স মোডে রাইট করা সিডির ক্ষেত্রে এটি নিয়ে আপনাকে ভাবে হবে না। `image.iso` নামে একটি ইমেজ সিডিতে রাইট করার পর চেকসাম পরীক্ষা করতে কমান্ডটি হবে:

```
me@howtocode-pc:~$ md5sum image.iso; md5sum /dev/sr0
```



কমান্ডটি দিলে, প্রথম লাইনে ইমেজের ও দ্বিতীয় লাইনে সিডির চেকসাম দেখাবে। যার ফলে আপনি সহজে মিলিয়ে দেখতে পারবেন।

কিন্তু ডিভিডি ডিস্কের ক্ষেত্রে আপনাকে আরেকটু কসরত করতে হবে। কমান্ডটি হবে এরকম:

```
me@howtocode-pc:~$ md5sum image.iso; dd if=/dev/sr0 bs=2048 count=$(( $(stat -c "%s" image.iso) / 2048 ))
```

রীতিমত ভয়ঙ্কর দেখাচ্ছে কি? আসুন, ভেঙে দেখা যাক:

- **md5sum image.iso;**: এটুকু দিয়ে আমরা image.iso এর চেকসাম দেখলাম।
- **dd**: আমরা dd কমান্ড ব্যবহার করছি ডিভিডিটি পড়তে।
- **if=/dev/sr0**: dd কমান্ডের ইনপুট হিসেবে আমরা ডিভিডির ডিভাইসটি দিয়েছি।
- **bs=2048**: আমরা dd কে একেকবারে 2048 বাইট করে পড়তে বলছি। সিডি/ডিভিডি সবসময় 2048 বাইটের ব্লকে রাইট করা হয়।
- **count=\$(( \$(stat -c "%s" image.iso) / 2048 ))**: আমরা কতগুলো ব্লক পড়তে হবে তা বলছি। এটি একটু জটিল। এর অংশগুলো এরকম:
  - **\$(stat -c "%s" image.iso)**: আমরা একটি কমান্ড সাবস্টিটিউশন ব্যবহার করছি। এই কমান্ডটি দিয়ে সম্পূর্ণ ইমেজটিকে বাইটের সাইজে দেখাবে। অর্থাৎ কমান্ডের এই অংশ একটি সংখ্যায় রূপান্তরিত হবে।
  - **=\$(( \$(stat -c "%s" image.iso) / 2048 ))**: এটি একটি গাণিতিক এক্সপ্রেশন। **\$(stat -c "%s" image.iso)** দিয়ে যে সংখ্যাটি পাওয়া যাবে সেটিকে 2048 দিয়ে ভাগ করবে। ফলে কাউন্টের জন্য সেই ব্লক সংখ্যার পরিমাণ পাওয়া যাবে।
- **| md5sum**: সবশেষে dd দিয়ে পড়া তথ্যের চেকসাম বের করতে আমরা প্রাপ্ত আউটপুটকে পাইপ দিয়ে md5sum এ পরিচালিত করেছি।

## ডিভাইস ক্লোনিং (Device Cloning)

আমরা সাধারণত ফাইলপত্র বা তথ্যকে সাজানো গোছানো দেখি ডিরেক্টরিতে। ফাইল বললেই চোখের সামনে এটাই ভেসে ওঠে। কিন্তু আপনি এটাকে ডাটাবেস হিসেবে দেখতে পারেন। স্টোরেজ ডিভাইসে ডাটা ব্লকের পর ব্লকে সাজানো থাকে। আপনি সাধারণভাবে কপি করলে একইভাবে না সাজানোর সম্ভবনাই বেশি। কিন্তু আপনার কাছে কোনো উপায় যদি থাকে যে একটি ডিভাইসে ব্লকগুলো যেভাবে সাজানো থাকে সেভাবে সাজাতে পারবেন তাহলে আপনি হুবহু ক্লোন করতে পারবেন।

এখন প্রশ্ন হচ্ছে এর কোনো ব্যবহার আদৌ আছে কিনা? আছে। এবং আপনারও দরকার হতে পারে কখনো কখনো। কখনো যদি কোনো লিনাক্সের ইমেজ ফাইল থেকে বুটাবল মিডিয়া বানাতে চান। এটা সবচেয়ে চমৎকার পদ্ধতি।

এরজন্য আমরা `dd` কমান্ড ব্যবহার করবো। মাথায় রাখবেন `dd` একটি বিধ্বংসী কমান্ড হতে পারে সামান্য ভুলে। অতএব সাবধান! `dd` কমান্ডের কমান্ড কাঠামো এরকম:

```
dd if=input_file of=output_file
```

`input_file` ও `output_file` দুটোই ফাইল বা ডিভাইস যেকোনোকিছু হতে পারে। মনে করি আপনি কম্পিউটারে দুটো পেনড্রাইভ লাগিয়েছেন। ডিভাইস দুটি এখন যথাক্রমে `sdb` ও `sdC`। এবং এদের মধ্য থেকে `sdb1` ও `sdC1` মাউন্টেড আছে। আমরা প্রথমে পার্টিশন দুটি আনমাউন্ট করবো এবং তারপর `sdb` থেকে `sdC` তে ক্লোন করবো:

```
# umount sdb1
# umount sdC1
# dd if=/dev/sdb of=/dev/sdC
```

আবার আমরা চাইলে `sdb` এর একটি ইমেজ ফাইল বানাতে পারি এভাবে:

```
# dd if=/dev/sdb of=sdb-backup.img
```

# ইমেজ তৈরী

তথ্য সংরক্ষণে অপটিক্যাল ড্রাইভ/ডিভাইস বা CD ও DVD এর ব্যবহার সম্প্রতি কমে এলেও বিলুপ্ত হয়নি। বিশেষ করে ব্যাকআপ রাখতে এর ব্যবহার প্রচুর। সিডি বা ডিভিডির হুবহু ক্লোনকে ইমেজ বলে। কয়েকরকম ইমেজ ফরম্যাটের মধ্যে সবচেয়ে জনপ্রিয় iso ফরম্যাটে। এই লেসনে আমরা ইমেজ নিয়ে বিভিন্ন কাজ শিখবো।

## সিডি বা ডিভিডি থেকে ইমেজ তৈরী করা

সিডি বা ডিভিডি থেকে আমরা `dd` কমান্ড দিয়ে ইমেজ তৈরী করতে পারি। ডিস্ক চোকানোর পর সাধারণত তার ডিভাইস নেম হয় `sr0`। সুতরাং এই ডিস্কটির `image.iso` নামে একটা ইমেজ বানাতে আমাদের লিখতে হবে:

```
# dd if=/dev/sr0 of=image.iso
```

এই পদ্ধতিতে আমরা ডাটা ডিস্কের ইমেজ তৈরী করতে হবে। তবে অডিও ডিস্কের ইমেজ তৈরী করতে গেলে আমাদের `cdrecord` ব্যবহার করতে হবে এভাবে:

```
# cdrecord read-cd --read-raw --datafile audio.bin --device /dev/sr0 --driver generic-mmc-r
```

এবার কমান্ডটির বিভিন্ন অংশ দেখা যাক:

- **cdrecord**: প্রোগ্রাম বা কমান্ডটির নাম।
- **read-cd**: কমান্ডটির বিভিন্ন মোড আছে। ইমেজ ফাইল তৈরী করতে আমরা `read-cd` মোড ব্যবহার করছি।
- **--read-raw**: ব্লক বাই ব্লক কপি করতে এটি ব্যবহার করা হয়।
- **--datafile audio.bin**: আমাদের ইমেজ ফাইলের নাম। অডিও ইমেজ এর এক্সটেনশন হয় `.bin`।
- **--device /dev/sr0**: আমাদের অডিও ডিস্কের ডিভাইস নাম।
- **--driver generic-mmc-raw**: সমস্ত কাজটি করার জন্য সিডিরমের কোন ড্রাইভার ব্যবহার করবে।
- **audio.toc**: ইমেজের ইনডেক্স ফাইলের নাম।

কমান্ডটি দেওয়ার পর আমরা বর্তমান ডিরেক্টরিতে `audio.bin` নামের একটি ইমেজ পাবো।

## ফাইল থেকে ইমেজ তৈরী

মনে করি, আমরা কিছু ফাইল থেকে একটি iso ইমেজ তৈরী করবো। প্রথমে ফাইলগুলো `~/image_content` ফোল্ডারে রাখলাম। এবার `genisoimage` কমান্ড দিয়ে আমরা `files.iso` নামে একটা ইমেজ বানাবো এভাবে:

```
me@howtocode-pc:~$ genisoimage -o files.iso -R -J ~/image_content
```

এবার কমান্ডটির বিভিন্ন অংশগুলো দেখা যাক:

- **genisoimage:** কমান্ডের নাম ।
- **-o files.iso:** -o অপশন দিয়ে আউটপুট ইমেজ ফাইলের নাম লিখেছি ।
- **-R -J:** -R ও -J অপশন যথাক্রমে Rock Ridge ও Joliet এক্সটেনসন ব্যবহার করে । যাদের জন্য ইউনিক্স ও উইন্ডোজে বড় ফাইলনেম রাখা যাবে ।
- **~/image-content:** যে ডিরেক্টরি থেকে তথ্য নিয়ে ইমেজ ফাইল বানাতে হবে ।

## অপটিক্যাল মিডিয়ায় রাইট করা

এই পর্যায়ে আমরা সিডি/ডিভিডি রাইট করা সম্পর্কিত বিভিন্ন কাজ দেখবো।

### রি-রাইটেবল সিডি/ডিভিডি ফাঁকা করা

আমরা অনেকেই ব্যাকআপ রাখতে রি-রাইটেবল সিডি/ডিভিডি ব্যবহার করি। এগুলো পুনর্ব্যবহারের জন্য আগে ফাঁকা করতে হয়। এজন্য আমরা এই কমান্ডটি দেবো:

```
me@howtocode-pc:~$ wodim dev=/dev/sr0 blank=fast
```

### ইমেজ ফাইল ডিভিডিতে রাইট করা

মনে করি আমাদের কাছে image.iso নামে একটি আইএসও ইমেজ আছে। এটিকে ফাঁকা সিডি/ডিভিডিতে রাইট করতে এই কমান্ডটি দিতে হবে:

```
me@howtocode-pc:~$ wodim dev=/dev/sr0 image.iso
```

## অধ্যায় - চার

## নেটওয়ার্কিং

নেটওয়ার্কিং এর প্রসঙ্গে বলা যায় যে, সম্ভবত নেটওয়ার্ক সম্পর্কিত এমন কোন কাজ নেই যা লিনাক্স দিয়ে করা যায় না। ফায়ারওয়াল, রাউটার, নেম সার্ভার, এনএএস বক্স এবং বিভিন্ন রকম নেটওয়ার্কিং সিস্টেম তৈরিতে সাধারণত লিনাক্স ব্যবহৃত হয়। নেটওয়ার্কিং সার্ভিসেস যেমন প্রচুর তেমনি নেটওয়ার্ক কনফিগার এবং কন্ট্রোল করার জন্য কমান্ডও প্রচুর। সবচেয়ে বেশি ব্যবহৃত কিছু কমান্ড এর প্রতি আমরা আলোকপাত করব। আমরা মূলত নেটওয়ার্ক মনিটরিং এবং ফাইল আদান-প্রদান করতে ব্যবহৃত কমান্ডগুলো আলোচনা করব। এছাড়াও ssh সম্পর্কে জানব। ssh মূলত রিমোট লগইন এ ব্যবহৃত হয়।

শুরু করার আগে নেটওয়ার্কিং এর খুব প্রাথমিক কিছু জিনিসের সহজবোধ্য ধারণাটা নিয়ে নেওয়া যাক।

## আইপি এড্রেস (IP Address)

আইপি এড্রেস হচ্ছে নেটওয়ার্কে ব্যবহৃত আপনার মূল ঠিকানা। এটি সংখ্যায় প্রকাশ করা হয়। আপনি যখন ইন্টারনেটে সংযুক্ত হন, আপনাকে একটি আইপি এড্রেস দেওয়া হয়। আইপি এড্রেস ইউনিক। অর্থাৎ প্রত্যেকের আইপি এড্রেস আলাদা আলাদা।

## হোস্টনেম (Hostname)

হোস্টনেম হচ্ছে নেটওয়ার্কের মধ্যে আপনার ডিভাইসটি যে নামে পরিচিত হবে সেটি। সাধারণত লোকাল সার্ভারে আপনার দেওয়া নামটিই ব্যবহৃত হয় তবে ওয়ার্ল্ড ওয়াইড ওয়েবে সাধারণত তারপর ডোমেইন নেম যুক্ত হয় বা ডোমেইনটিই হোস্টনেম হয়।

## ডোমেইন নেম (Domain Name)

সহজ করে বললে ওয়েবসাইটের যে ইউআরএল আপনি মনে রাখেন সেটিই ডোমেইন নেম। একটি ডোমেইন তখন কেবলমাত্র হোস্টনেম হতে পারে যখন তার জন্য একটি আইপি এড্রেস বরাদ্দ থাকে।

## ইউআরআই (URI)

ইন্টারনেটে কোনো একটা নির্দিষ্ট রিসোর্স যেমন কোনো ফাইল বা ছবি বা যেকোনোকিছুর ঠিকানা হচ্ছে URI। ইউআরআই দু'রকম। URL এবং URN। ইউআরএন হল কোনো নাম দিয়ে প্রকাশের ব্যবস্থা। এর ব্যবহার কম। যেমন আইএসবিএন এ প্রত্যেকটি বইয়ের একটি করে ইউআরএন আছে। ইউআরএল বেশ জনপ্রিয়। দৈনন্দিন কাজে এটিই ব্যবহার করি আমরা। ইন্টারনেট ব্রাউজ করার সময় প্রতিটা নতুন পেজ লোডের সাথে সাথে একটা করে ইউআরএল দেখতে পান এড্রেসবারে।

- **নেটওয়ার্ক পরীক্ষণ ও পর্যবেক্ষণ:** নেটওয়ার্ক পর্যবেক্ষণ ও পরীক্ষণে ব্যবহৃত কমান্ডসমূহ।

- ফাইল ট্রান্সফার: ftp, wget ও aria2 এর ব্যবহার।
- নিরাপদ যোগাযোগ: ssh, sftp ও scp এর ব্যবহার।

# নেটওয়ার্ক পরীক্ষণ এবং পর্যবেক্ষন

নাম দেখে ভয় পাওয়ার কিছু নাই। এই অংশে আপনাদের নেটওয়ার্কিং সম্পর্কিত কিছু কমান্ড দেখানো হবে যেগুলো ব্যবহার করে আপনি খুব সহজেই আপনার নেটওয়ার্ক সম্পর্কে জানতে পারবেন। সহজে ফাইল আদান প্রদান করতে পারবেন। সব থেকে বড় কথা আপনি একটা পরিষ্কার ধারণা পাবেন যে কিভাবে ইন্টারনেট কাজ করে।

## পিং (ping)

`ping` হচ্ছে সব থেকে বেসিক নেটওয়ার্কিং কমান্ড। এটি নেটওয়ার্ক হোস্ট এর কাছে একটি স্পেশাল রিকুয়েস্ট পাঠায়। `ping` কমান্ড এর সাথে একটি নেটওয়ার্ক হোস্ট দিতে হয়। মানে হচ্ছে `ping` কমান্ড দিতে হয় এইভাবে:

```
ping your_specified_network_host( website address, ip address )
```

`ping` কমান্ডটি আপনার নির্ধারিত নেটওয়ার্ক হোস্ট এর কাছে ICMP ECHO\_REQUEST নামক একটি বিশেষ নেটওয়ার্ক প্যাকেট পাঠায়। আপনার নেটওয়ার্ক ঠিক থাকলে আপনি মোটামুটি সব নেটওয়ার্ক ডিভাইস থেকে প্রতিউত্তর পাবেন। তো চলুন দেখা যাক আমাদের পিসি তে `ping` কমান্ড দিলে কি ঘটে।

আমি `ping` এর সাথে নেটওয়ার্ক হোস্ট হিসেবে `www.github.com` ব্যবহার করেছি। আপনি ইচ্ছা করলে অন্য সাইট এমনকি অন্য কারো আইপি এড্রেস ও ব্যবহার করতে পারেন।

```
me@howtocodepc:~$ ping www.github.com
PING github.com (192.30.252.130) 56(84) bytes of data.
64 bytes from github.com (192.30.252.130): icmp_seq=1 ttl=53 time=269 ms
64 bytes from github.com (192.30.252.130): icmp_seq=2 ttl=53 time=253 ms
64 bytes from github.com (192.30.252.130): icmp_seq=3 ttl=53 time=254 ms
64 bytes from github.com (192.30.252.130): icmp_seq=4 ttl=53 time=249 ms
64 bytes from github.com (192.30.252.130): icmp_seq=5 ttl=53 time=257 ms
64 bytes from github.com (192.30.252.130): icmp_seq=6 ttl=53 time=259 ms
64 bytes from github.com (192.30.252.130): icmp_seq=7 ttl=53 time=251 ms
64 bytes from github.com (192.30.252.130): icmp_seq=8 ttl=53 time=256 ms
64 bytes from github.com (192.30.252.130): icmp_seq=9 ttl=53 time=264 ms
64 bytes from github.com (192.30.252.130): icmp_seq=10 ttl=53 time=266 ms

--- github.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9011ms
```

একবার `ping` কমান্ড দেয়ার পর আপনি যতক্ষণ না ম্যানুয়ালি কমান্ড বন্ধ করবেন ততক্ষণ আপনার পিসি থেকে নির্দিষ্ট সময় পর পর( ডিফল্ট 1s ) প্যাকেট নেটওয়ার্ক হোস্ট এর কাছে যেতে থাকবে। `ping` কমান্ডটি যখন আপনি `ctrl + C` চেপে কমান্ডটিকে ইন্টারাপ্ট করবেন (আমার উদাহরণে ১০ম প্যাকেট ) তখন পর্যন্ত পারফরমেন্স এর পরিসংখ্যান দেখাবে। আপনার নেটওয়ার্ক যদি পারফেক্ট হয় তাহলে ০ পারসেন্ট প্যাকেট লস দেখাবে।



লিনাক্স হোস্ট সহ যেকোনো ডিভাইসকে আপনি এমনভাবে কনফিগার করতে পারবেন যেন তা ping রিকুয়েস্ট কে ইগনোর করে। এটি মূলত করা হয় নেটওয়ার্ক সিকিউরিটি নিশ্চিত করতে।

## রাউটার ট্রেসিং

একজন ইউজার যখন কোনো সার্ভারের সাথে যোগাযোগ করে তখন তাকে কয়েকটি রাউটার ঘুরে যেতে হয়। এই মধ্যবর্তী রাউটার সম্পর্কিত তথ্যের জন্য `traceroute` কমান্ডটি ব্যবহার করা হয়। আমরা যদি `gitbook.com` এর রাউটার ট্রেস করতে চাই তাহলে লিখতে হবে:

```
me@howtocode-pc:~$ traceroute gitbook.com
```

এবং আমরা এরকম একটি ফলাফল পেয়েছি:

```
me@howtocode-pc:~$ traceroute to gitbook.com (216.239.32.21), 30 hops max, 60 byte packet
 1  * * *
 2  10.26.206.193 (10.26.206.193)  99.166 ms  109.373 ms  119.094 ms
 3  10.26.206.244 (10.26.206.244)  139.325 ms  139.507 ms  139.705 ms
 4  202.4.173.34 (202.4.173.34)  149.646 ms  150.007 ms  159.432 ms
 5  103.7.249.69 (103.7.249.69)  80.005 ms  89.957 ms  90.246 ms
 6  xe-cig-010-jag-110.pico.net.bd (103.7.251.121)  90.441 ms  91.980 ms  82.035 ms
 7  xe-jig-120-cig-120.pico.net.bd (103.7.251.77)  71.758 ms  60.009 ms  59.706 ms
 8  103.7.251.138 (103.7.251.138)  79.648 ms  69.523 ms  69.430 ms
 9  103.7.249.250 (103.7.249.250)  159.567 ms  140.180 ms  140.004 ms
10  72.14.232.110 (72.14.232.110)  139.844 ms  149.779 ms  72.14.233.204 (72.14.233.204)
11  72.14.239.22 (72.14.239.22)  138.316 ms  148.782 ms  209.85.243.245 (209.85.243.245)
12  209.85.248.25 (209.85.248.25)  150.106 ms  72.14.239.61 (72.14.239.61)  130.583 ms  209
13  * * *
14  any-in-2015.1e100.net (216.239.32.21)  129.580 ms  139.720 ms  139.401 ms
```

আমরা দেখতে পাচ্ছি প্রথম লাইনে বলা হচ্ছে:

```
traceroute to gitbook.com (216.239.32.21), 30 hops max, 60 byte packets
```

অর্থাৎ `gitbook.com` যার IP address `216.239.32.21` এর জন্য রাউট খুঁজবে। সর্বোচ্চ ৩০টি হপ(hop) বা রাউটার পর্যন্ত সে আমলে নেবে এবং এর জন্য 60 বাইটের ডাটা প্যাকেট ব্যবহার করবে।

তারপরেই প্রতি লাইনে হপ এর নম্বরসহ একটি করে রেজাল্ট পাচ্ছি। যার শুরুতে সার্ভারের নাম তারপর ব্রাকেটে তার এড্রেস। যেক্ষেত্রে সার্ভার এসব তথ্য দিতে অস্বীকৃতি জানায় সেক্ষেত্রে আমরা `**` চিহ্ন দেখছি।

## নেটস্ট্যাট (netstat)

নেটওয়ার্ক সম্পর্কিত বিভিন্ন তথ্য আমরা নেটস্ট্যাট দিয়ে দেখতে পারি। যেমন বর্তমান নেটওয়ার্ক ইন্টারফেসগুলো দেখতে আমরা ব্যবহার করবো:

```
me@howtocode-pc:~$ netstat -ie
Kernel Interface table
eth0      Link encap:Ethernet  HWaddr 00:1e:33:97:47:c8
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:1007 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1007 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:160385 (160.3 KB)  TX bytes:160385 (160.3 KB)

ppp0      Link encap:Point-to-Point Protocol
          inet addr:10.128.191.139  P-t-P:10.64.64.64  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:2784 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3204 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:3
          RX bytes:1277912 (1.2 MB)  TX bytes:345401 (345.4 KB)

wlan0     Link encap:Ethernet  HWaddr 00:22:fa:1e:cd:44
          UP BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

এখানে আমরা কয়েকটি ইন্টারফেস দেখতে পাচ্ছি। এবং দেখতে পাচ্ছি তাদের সম্পর্কিত বিভিন্ন তথ্য। আমাদের ইন্টারফেসের নামের অর্থগুলো জানা দরকার কেননা পরবর্তীতে কাজে লাগবে। এখানকার চারটি ইন্টারফেস হল:

- **eth0**: ইথারনেট ইন্টারফেস।
- **lo**: লোকাল লুপব্যাক যা আপনাকে লোকালহোস্ট প্রোডাইড করে।
- **ppp0**: মোবাইল ব্রডব্যান্ড যা ইউএসবিতে কানেক্ট করেছে।
- **wlan0**: ওয়াইফাই ইন্টারফেস।

একই তথ্য আমরা `ifconfig` কমান্ড দিয়ে দেখতে পারি।

আমরা কার্নেল আইপি টেবিল দেখতে পারি এভাবে:

```
me@howtocode-pc:~$ netstat -r
```

```
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	MSS Window	irtt	Iface
default	10.64.64.64	0.0.0.0	UG	0 0	0	ppp0
10.64.64.64	*	255.255.255.255	UH	0 0	0	ppp0

## iftop

`iftop` কমান্ডের সাহায্যে আমরা সহজেই নেটওয়ার্ক ট্রাফিক দেখতে পারি। ডিফল্টভাবে এটি `eth0` ইন্টারফেসে কাজ করে। অর্থাৎ মোবাইল ব্রডব্যান্ড বা ওয়াইফাইতে কানেক্টেড থাকলে আপনাকে `-i` আর্গুমেন্ট দিয়ে ইন্টারফেসের নাম লিখতে হবে। যেমন ওয়াইফাই হলে লিখবো:

```
# iftop -i wlan0
```

# ফাইল ট্রান্সফার

ইন্টারনেটের বহুমুখী যে ব্যবহার আমরা দেখে থাকি তা গড়ে উঠেছে ফাইল ট্রান্সফারকে কেন্দ্র করে। আমাদের এই লেসনের আলোচ্য বিষয় ফাইল ট্রান্সফার এর বিভিন্ন উপায় এবং সেই উপায়গুলো যেখানে নিরাপত্তা কোনো বিবেচ্য বিষয় নয়। নিরাপদ ফাইল ট্রান্সফারের জন্য আমাদের পরের লেসন পর্যন্ত অপেক্ষা করতে হবে।

## এফটিপি (ftp)

এফটিপি এর পুরো অর্থ ফাইল ট্রান্সফার প্রোটোকল। বুঝতেই পারছেন, ফাইল ট্রান্সফারের জন্যই এর জন্ম। এর মাধ্যমে আপনি এফটিপি সার্ভারে এফটিপি ক্লায়েন্ট দিয়ে যুক্ত হয়ে ফাইল ডাউনলোড করতে পারেন। এর একটা সমস্যা হচ্ছে এটি ক্রেডেনশিয়াল টেক্সট হিসেবে ট্রান্সফার করে বলে নিরাপদ না। তাই অধিকাংশ এফটিপি সার্ভার 'anonymous' নামে একাউন্টে যেকোনো হিজিবিজি পাসওয়ার্ড দিয়ে ঢুকতে দেয়। এফটিপি ব্যবহারের জন্য আমরা ftp প্রোগ্রামটি ব্যবহার করবো।

আমরা এফটিপির মাধ্যমে mirror.dhakacom.com এ ঢুকবো:

```
me@howotocode-pc:~$ ftp mirror.dhakacom.com
Connected to mirror.dhakacom.com.
220 Welcome to dhakaCom FTP mirror server.
Name (mirror.dhakacom.com:me): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd ubuntu-releases/trusty
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 27 Feb 19 22:21 FOOTER.html
-rw-r--r-- 1 0 0 2943 Mar 10 05:59 HEADER.html
-rw-r--r-- 1 0 0 307 Mar 26 19:18 MD5SUMS
-rw-r--r-- 1 0 0 844 Feb 19 22:21 MD5SUMS-metalink
-rw-r--r-- 1 0 0 198 Feb 19 22:21 MD5SUMS-metalink.gpg
-rw-r--r-- 1 0 0 198 Mar 26 19:18 MD5SUMS.gpg
-rw-r--r-- 1 0 0 347 Mar 26 19:18 SHA1SUMS
-rw-r--r-- 1 0 0 198 Mar 26 19:18 SHA1SUMS.gpg
-rw-r--r-- 1 0 0 467 Mar 26 19:18 SHA256SUMS
-rw-r--r-- 1 0 0 198 Mar 26 19:18 SHA256SUMS.gpg
lrwxrwxrwx 1 0 0 41 Feb 20 02:18 ubuntu-14.04.2-desktop-amd64.iso
-rw-r--r-- 1 0 0 40180 Feb 19 22:17 ubuntu-14.04.2-desktop-amd64.iso.
lrwxrwxrwx 1 0 0 47 Feb 20 02:18 ubuntu-14.04.2-desktop-amd64.iso.
lrwxrwxrwx 1 0 0 42 Feb 20 02:18 ubuntu-14.04.2-desktop-amd64.list
lrwxrwxrwx 1 0 0 46 Feb 20 02:18 ubuntu-14.04.2-desktop-amd64.man
-rw-r--r-- 1 0 0 46233 Feb 19 22:21 ubuntu-14.04.2-desktop-amd64.meta
```

```

lrwxrwxrwx 1 0 0 40 Feb 20 02:18 ubuntu-14.04.2-desktop-i386.iso -
-rw-r--r-- 1 0 0 40459 Feb 19 22:17 ubuntu-14.04.2-desktop-i386.iso.t
lrwxrwxrwx 1 0 0 46 Feb 20 02:18 ubuntu-14.04.2-desktop-i386.iso.z
lrwxrwxrwx 1 0 0 41 Feb 20 02:18 ubuntu-14.04.2-desktop-i386.list
lrwxrwxrwx 1 0 0 45 Feb 20 02:18 ubuntu-14.04.2-desktop-i386.manif
-rw-r--r-- 1 0 0 45906 Feb 19 22:21 ubuntu-14.04.2-desktop-i386.metal
lrwxrwxrwx 1 0 0 40 Feb 20 02:18 ubuntu-14.04.2-server-amd64.iso -
-rw-r--r-- 1 0 0 24138 Feb 19 22:20 ubuntu-14.04.2-server-amd64.iso.t
lrwxrwxrwx 1 0 0 46 Feb 20 02:18 ubuntu-14.04.2-server-amd64.iso.z
lrwxrwxrwx 1 0 0 42 Feb 20 02:18 ubuntu-14.04.2-server-amd64.jigdo
lrwxrwxrwx 1 0 0 41 Feb 20 02:18 ubuntu-14.04.2-server-amd64.list
-rw-r--r-- 1 0 0 45905 Feb 19 22:21 ubuntu-14.04.2-server-amd64.metal
lrwxrwxrwx 1 0 0 45 Feb 20 02:18 ubuntu-14.04.2-server-amd64.templ
lrwxrwxrwx 1 0 0 39 Feb 20 02:18 ubuntu-14.04.2-server-i386.iso ->
-rw-r--r-- 1 0 0 23457 Feb 19 22:21 ubuntu-14.04.2-server-i386.iso.to
lrwxrwxrwx 1 0 0 45 Feb 20 02:18 ubuntu-14.04.2-server-i386.iso.zs
lrwxrwxrwx 1 0 0 41 Feb 20 02:18 ubuntu-14.04.2-server-i386.jigdo
lrwxrwxrwx 1 0 0 40 Feb 20 02:18 ubuntu-14.04.2-server-i386.list -
-rw-r--r-- 1 0 0 45578 Feb 19 22:21 ubuntu-14.04.2-server-i386.metali
lrwxrwxrwx 1 0 0 44 Feb 20 02:18 ubuntu-14.04.2-server-i386.templa
-rw-r--r-- 1 0 0 2551408 Apr 14 2014 wubi.exe
226 Directory send OK.
ftp> lcd ~/Desktop
Local directory now /home/me/Desktop
ftp> get MD5SUMS
local: MD5SUMS remote: MD5SUMS
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for MD5SUMS (307 bytes).
226 File send OK.
307 bytes received in 0.01 secs (30.8 kB/s)
ftp> exit
221 Goodbye.

```

এবার আসুন পুরো এফটিপি সেশনটির বিভিন্ন অংশ দেখি:

```

me@howotocode-pc:~$ ftp mirror.dhakacom.com
Connected to mirror.dhakacom.com.
220 Welcome to dhakaCom FTP mirror server.
Name (mirror.dhakacom.com:me): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.

```

প্রথমেই আমরা `ftp mirror.dhakacom.com` কমান্ড দিয়ে `mirror.dhakacom.com` এ যুক্ত হয়েছি। তারপর আমাদের নাম অর্থাৎ ইউজারনেম জানতে চেয়েছে। আমরা দিয়েছি **anonymous** এবং পাসওয়ার্ড জানতে চাইলে হিজিবিজি যা ইচ্ছে তাই পাসওয়ার্ড দিয়েছি। এবং দেখাচ্ছে যে আমরা সাফল্যে লগিন করেছি। এরপর আমাদের জন্য **ftp>** প্রম্পট এসেছে।

```

ftp> cd ubuntu-releases/trusty
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 27 Feb 19 22:21 FOOTER.html
-rw-r--r-- 1 0 0 2943 Mar 10 05:59 HEADER.html
-rw-r--r-- 1 0 0 307 Mar 26 19:18 MD5SUMS
-rw-r--r-- 1 0 0 844 Feb 19 22:21 MD5SUMS-metalink
-rw-r--r-- 1 0 0 198 Feb 19 22:21 MD5SUMS-metalink.gpg
-rw-r--r-- 1 0 0 198 Mar 26 19:18 MD5SUMS.gpg
-rw-r--r-- 1 0 0 347 Mar 26 19:18 SHA1SUMS
-rw-r--r-- 1 0 0 198 Mar 26 19:18 SHA1SUMS.gpg
-rw-r--r-- 1 0 0 467 Mar 26 19:18 SHA256SUMS
-rw-r--r-- 1 0 0 198 Mar 26 19:18 SHA256SUMS.gpg
lrwxrwxrwx 1 0 0 41 Feb 20 02:18 ubuntu-14.04.2-desktop-amd64.iso
-rw-r--r-- 1 0 0 40180 Feb 19 22:17 ubuntu-14.04.2-desktop-amd64.iso.
lrwxrwxrwx 1 0 0 47 Feb 20 02:18 ubuntu-14.04.2-desktop-amd64.iso.
lrwxrwxrwx 1 0 0 42 Feb 20 02:18 ubuntu-14.04.2-desktop-amd64.list
lrwxrwxrwx 1 0 0 46 Feb 20 02:18 ubuntu-14.04.2-desktop-amd64.manif
-rw-r--r-- 1 0 0 46233 Feb 19 22:21 ubuntu-14.04.2-desktop-amd64.meta
lrwxrwxrwx 1 0 0 40 Feb 20 02:18 ubuntu-14.04.2-desktop-i386.iso -
-rw-r--r-- 1 0 0 40459 Feb 19 22:17 ubuntu-14.04.2-desktop-i386.iso.t
lrwxrwxrwx 1 0 0 46 Feb 20 02:18 ubuntu-14.04.2-desktop-i386.iso.z
lrwxrwxrwx 1 0 0 41 Feb 20 02:18 ubuntu-14.04.2-desktop-i386.list
lrwxrwxrwx 1 0 0 45 Feb 20 02:18 ubuntu-14.04.2-desktop-i386.manif
-rw-r--r-- 1 0 0 45906 Feb 19 22:21 ubuntu-14.04.2-desktop-i386.metal
lrwxrwxrwx 1 0 0 40 Feb 20 02:18 ubuntu-14.04.2-server-amd64.iso -
-rw-r--r-- 1 0 0 24138 Feb 19 22:20 ubuntu-14.04.2-server-amd64.iso.t
lrwxrwxrwx 1 0 0 46 Feb 20 02:18 ubuntu-14.04.2-server-amd64.iso.z
lrwxrwxrwx 1 0 0 42 Feb 20 02:18 ubuntu-14.04.2-server-amd64.jigdo
lrwxrwxrwx 1 0 0 41 Feb 20 02:18 ubuntu-14.04.2-server-amd64.list
-rw-r--r-- 1 0 0 45905 Feb 19 22:21 ubuntu-14.04.2-server-amd64.metal
lrwxrwxrwx 1 0 0 45 Feb 20 02:18 ubuntu-14.04.2-server-amd64.templ
lrwxrwxrwx 1 0 0 39 Feb 20 02:18 ubuntu-14.04.2-server-i386.iso ->
-rw-r--r-- 1 0 0 23457 Feb 19 22:21 ubuntu-14.04.2-server-i386.iso.to
lrwxrwxrwx 1 0 0 45 Feb 20 02:18 ubuntu-14.04.2-server-i386.iso.zs
lrwxrwxrwx 1 0 0 41 Feb 20 02:18 ubuntu-14.04.2-server-i386.jigdo
lrwxrwxrwx 1 0 0 40 Feb 20 02:18 ubuntu-14.04.2-server-i386.list -
-rw-r--r-- 1 0 0 45578 Feb 19 22:21 ubuntu-14.04.2-server-i386.metal
lrwxrwxrwx 1 0 0 44 Feb 20 02:18 ubuntu-14.04.2-server-i386.templa
-rw-r--r-- 1 0 0 2551408 Apr 14 2014 wubi.exe
226 Directory send OK.

```

আমরা প্রথমে `cd ubuntu-releases/trusty` দিয়ে **ubuntu-releases/trusty** ফোল্ডারে ঢুকেছি। এখানে উবুন্টু ১৪.০৪ এর ইমেজ ও অন্যান্য ফাইল আছে। আমরা **ls** কমান্ড দিয়ে ফাইলগুলোর একটি লিস্ট পেয়েছি।

```
ftp> lcd ~/Desktop
Local directory now /home/me/Desktop
ftp> get MD5SUMS
local: MD5SUMS remote: MD5SUMS
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for MD5SUMS (307 bytes).
226 File send OK.
307 bytes received in 0.01 secs (30.8 kB/s)
```

আমরা প্রথমে `lcd ~/Desktop/` কমান্ড দিয়ে আমাদের লোকাল ডিরেক্টরি `~/Desktop` করে নিলাম। অর্থাৎ, যে ফাইলটি ডাউনলোড করবো তা ডেস্কটপে থাকবে। তারপর আমরা MD5SUMS নামের ফাইলটি ডাউনলোড করতে `get MD5SUMS` কমান্ডটি দিয়েছি।

```
ftp> exit
221 Goodbye.
```

সবশেষে `exit` কমান্ড দিয়ে এফটিপি সংযোগ বন্ধ করেছি।

আমরা ftp এর বদলে lftp ও ব্যবহার করতে পারতাম। এটি anonymous একাউন্টে অটোমেটিক লগিন করে। ট্যাব কম্পিলেশন এবং অটোমেটিক রিট্রাইও সাপোর্ট করে।

## wget

`wget` একটি মাল্টিপ্রোটোকল ডাউনলোড ম্যানেজার। এর কমান্ড কাঠামো:

```
wget file_url
```

যেমন `howtocode.com.bd` এর প্রথম পাতা আমরা ডাউনলোড করতে পারি `wget http://howtocode.com.bd/index.html` কমান্ড দিয়ে। `wget` দিয়ে ডাউনলোড করার সময় আমরা `ctrl-c` চেপে ডাউনলোড বন্ধ করতে পারি। পরবর্তীতে সেই ডাউনলোড চালু করতে `-c` অপশন যোগ করতে হয়। প্রথম ডাউনলোডের সময়ও যদি `-c` ব্যবহার করেন তো ক্ষতি নেই। তাই সবসময়ই আপনি এভাবে ডাউনলোড করতে পারেন:

```

me@howtocode-pc:~$ wget -c http://howtocode.com.bd/index.html
--2015-03-31 13:21:32-- http://howtocode.com.bd/index.html
Resolving howtocode.com.bd (howtocode.com.bd)... 192.30.252.153, 192.30.252.154
Connecting to howtocode.com.bd (howtocode.com.bd)|192.30.252.153|:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: /index.html [following]
--2015-03-31 13:21:34-- http://howtocode.com.bd/index.html
Connecting to howtocode.com.bd (howtocode.com.bd)|192.30.252.153|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: http://www.howtocode.com.bd/index.html [following]
--2015-03-31 13:21:34-- http://www.howtocode.com.bd/index.html
Resolving www.howtocode.com.bd (www.howtocode.com.bd)... 199.27.75.133
Connecting to www.howtocode.com.bd (www.howtocode.com.bd)|199.27.75.133|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 45059 (44K) [text/html]
Saving to: 'index.html'

100%[=====>] 45,059      61.2KB/s   in 0.7s

2015-03-31 13:21:36 (61.2 KB/s) - 'index.html' saved [45059/45059]

```

## aria2

এটি আমার দেখা সবচেয়ে দারুণ ডাউনলোড ম্যানেজার। এর অনেক অনেক ফাংশনালিটি। এটি পাবেন aria2 প্যাকেজে এবং কমান্ডটি হল `aria2c`। এর আর্গুমেন্ট হিসেবে আপনি লিঙ্ক দিতে পারেন, টরেন্ট ফাইল বা ম্যাগনেট লিঙ্ক দিতে পারেন। এমনকি একই ফাইলের যদি একাধিক ডাউনলোড লিঙ্ক, টরেন্ট লিঙ্ক ইত্যাদি যদি থাকে আপনি আর্গুমেন্ট হিসেবে সবগুলো যোগ করলে সবজায়গা থেকে ডাউনলোড করে একটি ফাইল হিসেবে সেভ করতে পারে।



## নিরাপদ যোগাযোগ

একটি কম্পিউটারকে দূরবর্তী আরেকটি কম্পিউটার দিয়ে নিয়ন্ত্রণ করা ইউনিক্স সিস্টেমের জন্য নতুন কোনো ব্যাপার না। আগে এই কাজে rlogin বা telnet ব্যবহৃত হত। কিন্তু সমস্যা হচ্ছে যে, এগুলো এফটিপি'র মত প্লেইনটেক্সটে ডাটা ট্রান্সফার করত। যার ফলে এগুলো ইন্টারনেটের ভিতরে ব্যবহার করা অনিরাপদ হয়ে উঠল।

### ssh

এই সমস্যা থেকে সমাধান দিল ssh বা secure shell। এই প্রযুক্তির দুটো অংশ। একটি হল ssh server অন্যটি ssh client।

### ssh সার্ভার তৈরী

এর জন্য আমরা প্রথমে openssh-server প্যাকেজটি ইন্সটল করবো। এবার আমরা কনফিগার করবো। কনফিগার করতে আমাদের /etc/ssh/sshd\_config ফাইলটি এডিট করতে হবে।। এর জন্য আমরা এর একটি ব্যাকআপ রাখবো এভাবে:

```
# cp /etc/ssh/sshd_config /etc/ssh/sshd_config.factory-defaults
```

এবার ন্যানো, ভিম বা ইম্যাকস দিয়ে ফাইলটি রুটমোডে খুলবো:

```
# emacs -nw cp /etc/ssh/sshd_config
```

ফাইলে যেসব লাইনগুলো # দিয়ে শুরু সেগুলো কমেন্ট। আমরা একলাইনে দেখতে পাচ্ছি:

```
Port 22
```

এটি ssh এর ডিফল্ট পোর্ট। এজন্য ব্যবহার করাও অনিরাপদ। আমরা এটিকে 365 তে পরিবর্তিত করতে এডিট করে এটি লিখবো:

```
Port 365
```

এবার আমরা সেভ করে বন্ধ করতে পারি।

এরপর আমরা ssh সার্ভার রিস্টার্ট দেবো এভাবে:

```
# /etc/init.d/ssh restart
```

## ssh লগইন

লগইন করতে আপনাকে সার্ভারের হোস্টনেম জানতে হবে। আপনার কম্পিউটারে আপনি নিজে localhost ব্যবহার করে ঢুকতে পারেন। লোক্যাল নেটওয়ার্কের অন্য কম্পিউটারের হোস্টনেম ব্যবহার করে ঢুকতে পারেন। ইন্টারনেটের ক্ষেত্রে হয় আপনাকে তার ওয়েব এড্রেস অথবা আইপি এড্রেস জানতে হবে।

আমরা অধিকাংশই ডেডিকেটেড আইপি ব্যবহার করিনা। ফলে প্রত্যেক কানেকশনের সময় আইপি বদলায়। তাই আমাদের আইপি জানতে হবে যখন রিমোটলি এক্সেস করতে চাই। ধরুন, আমি চাই কাউকে আমার কম্পিউটারে এক্সেস দিতে। তাহলে আমি তাকে পাবলিক আইপি জানাবো। এটা জানতে এই কমান্ডটি দিতে পারি:

```
curl -s checkip.dyndns.org | sed -e 's/.*Current IP Address: //' -e 's/<.*$//'
```

আপনি এটি alias করে রাখতে পারেন।

এখন আমি আমার কম্পিউটারে যুক্ত হবো। me নামের একজন ইউজার হিসেবে। আমরা যেহেতু ডিফল্ট পোর্টটি ব্যবহার না করে 365 ব্যবহার করছি তাই আমাদের কমান্ড হবে:

```
me@howtocode-pc:~$ ssh -p 365 me@localhost
```

আমরা -p অপশন দিয়ে port লিখেছি। তারপর me@localhost দিয়ে localhost এর me এর সাথে যোগাযোগ করছি। এরপর আমরা এমনকিছু দেখতে পাবো:

```
The authenticity of host '[localhost]:365 ([127.0.0.1]:365)' can't be established.  
ECDSA key fingerprint is 22:a4:cf:1a:e3:d9:3f:ae:fa:ca:ab:8b:5a:8e:64:01.  
Are you sure you want to continue connecting (yes/no)? yes  
Warning: Permanently added '[localhost]:365' (ECDSA) to the list of known hosts.  
me@localhost's password:  
Welcome to Ubuntu 14.04.2 LTS (GNU/Linux 3.13.0-48-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com/
```

```
The programs included with the Ubuntu system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.
```

```
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by  
applicable law.
```

```
me@howtocode-pc:~$ exit  
logout  
Connection to localhost closed.
```

প্রথমবার লগিনের সময় অথেনটিকেট করা সম্ভব হয়না। তারবদলে একটি ফিঙ্গারপ্রিন্ট দেখায়। এটি যদি ঠিক থাকে তাহলে আপনি yes লিখে এন্টার দেবেন। ফিঙ্গারপ্রিন্ট আপনাকে সার্ভারের এডমিনিস্ট্রেটর সরবরাহ করবে। তারপর আপনাকে me ইউজারের পাসওয়ার্ড দিতে হবে। সঠিকভাবে দিলে আপনাকে প্রম্পট দিবে। তখন আপনি সাধারণ টার্মিনালের মতই কাজ করতে পারবেন। সেশন বন্ধ করতে `exit` লিখে এন্টার দেবেন।

## ssh-key এর মাধ্যমে লগইন

লগইনের আরেকটি উপায় হল ssh key তৈরী করা। এর দুটি অংশ। একটি পাবলিক কী, অন্যটি প্রাইভেট। ইউজার তথ্য প্রাইভেট কী দিয়ে এনক্রিপ্ট করে দিলে সেটি পাবলিক কী দিয়ে ডিক্রিপ্ট করা যায়। কিন্তু তার পরিবর্তন করা যায় না। এজন্য এটি বহুলব্যবহৃত একটি ব্যবস্থা। এজন্য আপনাকে একজোড়া কী জেনারেট করতে হবে। কী জেনারেট করতে পারেন এভাবে:

```
me@howtocode-pc:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/me/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/me/.ssh/id_rsa.
Your public key has been saved in /home/me/.ssh/id_rsa.pub.
The key fingerprint is:
be:eb:32:90:38:74:1a:65:b8:83:3a:64:7d:b1:84:33 me@howtocode-pc
The key's randomart image is:
+--[ RSA 2048 ]-----+
|  ..                |
| .Eoo              |
| ..++ o           |
|.0=..o            |
|+. *.. S          |
|o + o .           |
| . . . .          |
|      o .         |
|      ++.         |
+-----+

```

আপনি `ssh-keygen` কমান্ডটি দিলে প্রথমে জিজ্ঞাসা করবে কোন ফাইলে সেভ করবে। আপনি ডিফল্টটি সিলেক্ট করতে এন্টার চাপুন। এরপর পাসফ্রেজ/পাসওয়ার্ড দিতে বলবে আপনি নতুন একটি পাসফ্রেজ দুবার দিলে কী তৈরী হবে।

এবার আপনি নতুন তৈরী কী সিস্টেমে যোগ করবেন এভাবে:

```
me@howtocode-pc:~$ ssh-add
Enter passphrase for /home/me/.ssh/id_rsa:
Identity added: /home/me/.ssh/id_rsa (/home/me/.ssh/id_rsa)

```

`ssh-add` কমান্ডটি দেওয়ার পর পাসফ্রেজ জানতে চাইবে। সঠিক পাসফ্রেজ দিলে কীটি যুক্ত হবে।

এবার আপনার পাবলিক কীটি সার্ভারে পাঠানোর পালা। এটি করবেন এভাবে:

```
me@howtocode-pc:~$ ssh-copy-id -p 365 -i me@localhost
```

এরপর আপনি যে ইউজার হিসেবে লগিন করছেন তার পাসওয়ার্ড জিজ্ঞাসা করবে। সঠিক পাসওয়ার্ড দুলে কীটি যুক্ত হবে। এরপর থেকে লগিনের সময় আর কোনোরকমের অথেনটিকেশনের দরকার হবে না।

কখনো কখনো এমনও হতে পারে যে আপনি মাত্র একটা কমান্ড দিতে চাচ্ছেন রিমোটলি। তার জন্য পুরো লগইন এর প্রয়োজন নেই। যেমন আমরা যদি রিমোট কম্পিউটারের র‍্যাম ও সোয়াপ সম্পর্কিত তথ্য জানতে 'free' কমান্ডটি ব্যবহার করতে চাই তাহলে সেটি করতে পারি এভাবে:

```
me@howtocode-pc:~$ ssh -p 365 me@localhost free
```

	total	used	free	shared	buffers	cached
Mem:	3915316	1607456	2307860	182148	150292	670024
-/+ buffers/cache:		787140	3128176			
Swap:	2097148	0	2097148			

আবার আমরা যদি চাই এই কমান্ডের আউটপুট আমাদের লোকাল কম্পিউটারে memlog.txt ফাইলে রিডিং করতে চাই তাহলে লিখবো:

```
me@howtocode-pc:~$ ssh -p 365 me@localhost 'free' > memlog.txt
```

এখানে free কে " আবদ্ধ করা জরুরি। এরকমক্ষেত্রে আবদ্ধ অংশটুকু রিমোট কম্পিউটারে এবং আবদ্ধ নয় এমন অংশ লোকাল কম্পিউটারে কাজ করবে। অতএব আমরা যদি memlog.txt কে 'memlog.txt' লিখতাম তাহলে টেক্সটফাইলটি রিমোট কম্পিউটারে তৈরী হত।

## scp ও sftp

scp অনেকটা cp কমান্ডের মতই তবে এটি রিমোটলি কাজ করতে পারে। অর্থাৎ আপনি যদি লোকাল কম্পিউটারের সাথে রিমোট কম্পিউটারে তথ্য আদানপ্রদান করতে চান তাহলে এটি ব্যবহার করতে পারেন। নীচে দুটি উদাহরণ দেওয়া হল:

```
me@howtocode-pc:~$ scp -P 365 to_remote.txt me@localhost:./
to_remote.txt                                100% 331    0.3KB/s   00:00
me@howtocode-pc:~$ scp -P 365 me@localhost:from_remote.txt ./
from_remote.txt                              100% 331    0.3KB/s   00:00
```

প্রথম উদাহরণে আমরা লোকাল কম্পিউটারের হোম থেকে to\_remote.txt ফাইলটি রিমোট কম্পিউটারের হোমে কপি করেছি। দ্বিতীয় উদাহরণে from\_remote.txt ফাইলটি রিমোট থেকে লোকাল কম্পিউটারে। রিমোট কম্পিউটারের ফাইল বোঝাতে আমরা me@localhost: প্রেফিক্স ব্যবহার করছি।

`ssh` এর আরেকটি ফিচার হচ্ছে `sftp` অর্থাৎ সিকিউর এফটিপি। এটি এফটিপির মতই তবে নিরাপদ। আমরা রিমোট কম্পিউটারে `sftp` দিয়ে যোগাযোগ করতে পারি এভাবে:

```
me@howtocode-pc:~$ sftp -P 365 utsargo@localhost
Connected to localhost.
sftp>
```

## অধ্যায় - পাঁচ

# ফাইল সার্চ

এই অধ্যায় এর বিষয়বস্তু ফাইল সার্চ করা। হ্যাঁ, আপনি ফাইল ম্যানেজার চালু করে ফাইলের নাম লিখে সার্চ করতেই পারেন কিন্তু আবারো, কমান্ডলাইন দেবে ফ্লেকজিবিলিটি।

- **Locate:** নাম দিয়ে ফাইল সার্চ: locate কমান্ডের ব্যবহার।
- **find:** শক্তিশালী সার্চ: find কমান্ডের ব্যবহার।
  - **find: টেস্ট:** টেস্ট সম্পর্কিত ধারণা।
  - **find: অপারেটর:** অপারেটর সম্পর্কিত ধারণা।
  - **find: একশন:** একশন ও xargs এর ব্যবহার।
  - **find: অপশন:** find এর কিছু অপশন।
  - **অনুশীলন:** find সংক্রান্ত অনুশীলন।

## Locate: নাম দিয়ে ফাইল সার্চ

`locate` বোধহয় সবচেয়ে সোজাসাপ্টা ফাইল সার্চিং কমান্ড। হাতে-কীবোর্ডে দেখা যাক। আমরা সেইসব ফাইল খুঁজবো যেগুলোর মধ্যে **zip** কথাটি আছে:

```
me@howtocode-pc:~$ locate zip
```

আপনি লম্বা একটা লিস্ট দেখতে পাবেন। এখন মনে করুন আপনি চাইছেন সেইসব ফাইল খুঁজতে যার শুরু **zip** দিয়ে। যদি কোনো ফাইলের নামের শুরু **zip** দিয়ে হয় তবে তার শুরুতে অবশ্যই **/** থাকবে। অতএব আমরা লিখবো:

```
me@howtocode-pc:~$ locate /zip
```

কমান্ডের ফলাফলের একাংশ:

```
/usr/bin/zip
/usr/bin/zipcloak
/usr/bin/zipdetails
/usr/bin/zipgrep
/usr/bin/zipinfo
/usr/bin/zipnote
/usr/bin/zipsplit
/usr/lib/jvm/java-7-openjdk-amd64/jre/lib/ext/zipfs.jar
/usr/lib/python2.7/zipfile.py
/usr/lib/python2.7/zipfile.pyc
/usr/lib/python2.7/dist-packages/Pillow-2.3.0.egg-info/zip-safe
/usr/lib/python2.7/dist-packages/bzrlib/export/zip_exporter.py
/usr/lib/python2.7/dist-packages/bzrlib/export/zip_exporter.pyc
/usr/lib/python2.7/dist-packages/deluge/plugins/AutoAdd-1.04.egg/EGG-INFO/zip-safe
/usr/lib/python2.7/dist-packages/deluge/plugins/Blocklist-1.2.egg/EGG-INFO/zip-safe
/usr/lib/python2.7/dist-packages/deluge/plugins/Execute-1.2.egg/EGG-INFO/zip-safe
/usr/lib/python2.7/dist-packages/deluge/plugins/Extractor-0.2.egg/EGG-INFO/zip-safe
/usr/lib/python2.7/dist-packages/deluge/plugins/Label-0.2.egg/EGG-INFO/zip-safe
```

কিন্তু মনে করি আমরা আরো স্পেসিফিক জায়গায় খুঁজতে চাই। **bin** ফোল্ডারের মধ্যে **zip**। তাহলে লিখবো:

```
me@howtocode-pc:~$ locate bin/zip
/usr/bin/zip
/usr/bin/zipcloak
/usr/bin/zipdetails
/usr/bin/zipgrep
/usr/bin/zipinfo
/usr/bin/zipnote
/usr/bin/zipsplit
```

এবার আমরা এর সম্ভব ফাইলগুলো পেয়েছি।

এবার আরেকটু জটিল কিছু চেষ্টা করি। আমরা এমন কিছু ফাইল খুঁজবো যার শুরু **zip** দিয়ে এবং কোনো না কোনোভাবে সেটি python3 এর সাথে সম্পর্কযুক্ত। এজন্য আমরা এভাবে কমান্ড করবো:

```
me@howtocode-pc:~$ locate /zip | grep python3
/usr/lib/python3/dist-packages/pip/commands/zip.py
/usr/lib/python3/dist-packages/pip/commands/__pycache__/zip.cpython-34.pyc
/usr/lib/python3/dist-packages/setuptools-3.3.egg-info/zip-safe
/usr/lib/python3.4/zipfile.py
/usr/lib/python3.4/__pycache__/zipfile.cpython-34.pyc
/usr/local/lib/python3.4/dist-packages/django/views/decorators/gzip.py
/usr/local/lib/python3.4/dist-packages/django/views/decorators/__pycache__/gzip.cpython-3
```

আমরা যা করেছি তা হলো প্রথম `locate /zip` দিয়ে **zip** দিয়ে শুরু হওয়া সব ফাইল পেয়েছি। তারপর পাইপ এর সাহায্যে `grep python3` যোগ করেছি। ফলে `grep locate` এর আউটপুটকে ফিল্টার করে শুধু যেসব ফাইলপাথে python3 কথাটি আছে সেগুলো দেখিয়েছে।

## updatedb

`locate` কমান্ডটি সার্চের জন্য একটি ডাটাবেজ ব্যবহার করে। এই ডাটাবেজ ক্রনজবের মাধ্যমে নির্দিষ্ট সময় পরপর আপডেট হয়। তাই খুব একটা আগে তৈরী হয়নি এমন ফাইল আপনি নাও পেতে পারেন `locate` কমান্ড দিয়ে। এক্ষেত্রে আপনি সুপারইউজার মোডে `updatedb` কমান্ড দিলে ডাটাবেজ আপডেটেড হবে।



## find: শক্তিশালী সার্চ

`locate` যেখানে ডাবাবেজের ওপর নির্ভর করে ফাইলনেম/ফাইলপাথের ওপর সার্চ চালায়, `find` সেখানে কিছু টেস্ট কেসের ভিত্তিতে একটি ডিরেক্টরি ও তার সাবডিরেক্টরির মধ্য থেকে ফাইল খুঁজে বের করে। ফাইন্ডের আর্গুমেন্ট হিসেবে অবশ্যই দিতে হবে একটি ডিরেক্টরি। যেমন আমরা যদি হোম ফোল্ডারে `find` কমান্ড দিতে চাই তাহলে কমান্ডটি হবে:

```
me@howtocode-pc: find ~
```

স্বাভাবিকভাবেই বিশাল লিস্ট দেখাবে। তারবদলে আমরা যদি নীচের কমান্ডটি দিই তাহলে মোট ফাইল ও ডিরেক্টরির সংখ্যাটি পাবো:

```
me@howtocode-pc: find ~ | wc -l
179904
```

এতক্ষণ যা করলাম তা মোটেও কোনো কাজে আসবে না সম্ভবত। `find` কমান্ডটিকে কাজে লাগাতে গেলে আপনাকে টেস্ট (test) ও অপারেটর (Operator) সম্পর্কে জানতে হবে।

- **find: টেস্ট:** টেস্ট সম্পর্কিত ধারণা।
- **find: অপারেটর:** অপারেটর সম্পর্কিত ধারণা।
- **find: একশন:** একশন ও xargs এর ব্যবহার।
- **find: অপশন:** find এর কিছু অপশন।
- **অনুশীলন:** find সংক্রান্ত অনুশীলন।

## টেস্ট (test)

`find` প্রোগ্রামটিতে অপশন হিসেবে আমরা কিছু ফিল্টার যোগ করতে পারি। অর্থাৎ, সার্চ করার সময় ওই বিষয়গুলোর প্রতি নজর দেবে। এবং সেইসব ফাইলই দেখাবে যেগুলো শর্তপূরণ করে। মোটের ওপর সবরকম টেস্টকে আমরা তিনভাবে ভাগ করতে পারি।

- ফাইলের ধরণ সংক্রান্ত টেস্ট: এই টেস্টগুলো দিয়ে ফাইলের ধরণ নির্দিষ্ট করে দেওয়া যায়।
- ফাইলের আকার সংক্রান্ত টেস্ট: এগুলো ব্যবহৃত হয় নির্দিষ্ট আকারের ফাইল খুঁজতে।
- অন্যান্য: উপরের দুই ধরণের ছাড়া বাকি সব টেস্ট এই ধরণের অন্তর্ভুক্ত।

### ফাইলের ধরণ সংক্রান্ত টেস্ট

`find ~ | wc -l` কমান্ড দিয়ে আমরা হোম ডিরেক্টরির সকল ফাইল ও সাবডিরেক্টরির মিলিত সংখ্যা পেয়েছিলাম। এখন আমরা যদি চাই শুধু ডিরেক্টরির সংখ্যা পেতে তাহলে কী করতে হবে? আমাদের `find` প্রোগ্রামকে বলতে হবে যে শুধু ডিরেক্টরিগুলোই যেন সে আমলে আনে। আমরা সেটি করতে পারি এভাবে:

```
me@howtocode-pc: find ~ -type d | wc -l
13840
```

তাহলে, আমরা দেখতে পাচ্ছি, কোনো নির্দিষ্ট ধরণের ফাইল খুঁজতে হলে আমাদের **-type** অপশনটি যোগ করতে হবে এবং তার সাথে দিতে হবে ফাইলের ধরণ নির্দেশক বিশেষ চিহ্ন। যেমন এক্ষেত্রে **d** দিয়ে directory বোঝানো হয়েছে। আমরা নিচের টেবিল থেকে যেকোনো চিহ্ন ধরণ বোঝাতে ব্যবহার করতে পারি:

চিহ্ন	ফাইলের ধরণ
b	ব্লক স্পেশাল ডিভাইস ফাইল
c	ক্যারেঙ্চার স্পেশাল ডিভাইস ফাইল
d	ডিরেক্টরি
f	সাধারণ ফাইল
l	সিম্বোলিক লিঙ্ক

### ফাইলের আকার সংক্রান্ত টেস্ট

এবার মনে করুন আমরা সেইসব ফাইল সংখ্যা জানতে চাই যেগুলোর আকার ৫০০ মেগাবাইটের চেয়ে বড়। তাহলে আমরা সেটা করতে পারি এই কমান্ড দিয়ে:

```
me@howtocode-pc: find ~ -type f -size +500M | wc -l
11
```

তারমানে আকার অনুযায়ী ফিল্টার করতে আমাদের **-size** অপশন ব্যবহার করতে হবে। তারপর '+' বা '-' দিয়ে বড় বা ছোট বোঝাতে হবে। তারপর সংখ্যা ও একক নির্দেশক চিহ্ন। যেমন এখানে মেগাবাইটের জন্য **M** ছিল।

এবার আমরা চিহ্নগুলো দেখে নিই:

চিহ্ন	একক
b	৫১২ বাইটের ব্লক। কোনো এককনির্দেশক চিহ্ন না থাকলে এটি ডিফল্ট হিসেবে ব্যবহৃত হয়।
c	বাইট
w	২ বাইটের শব্দ
k	কিলোবাইট
M	মেগাবাইট
G	গিগাবাইট

## অন্যান্য টেস্ট

মনে করি হোম ফোল্ডারে আমরা কিছু ISO ইমেজ খুঁজবো যেগুলো ৫০০এমবির চেয়ে বড়। যেহেতু ইমেজগুলো ফাইল এবং এর এক্সটেনসন হবে **iso** আমরা লিখতে পারি:

```
me@howtocode-pc: find ~ -type f -name "*.iso" -size +500M
/home/me/porteus-refinished.iso
/home/me/lubuntu-14.04.1-desktop-i386.iso
/home/me/porteus-pocketboy-25-12-14.iso
/home/me/archlinux-2015.01-1-archboot.iso
/home/me/debian-7.7.0-amd64-CD-1.iso
/home/me/lubuntu-14.10-desktop-i386.iso
```

এখানে আমরা **-name** অপশন যোগ করেছি। আর ওয়াইল্ডকার্ড ব্যবহার করে সেইসব ফাইল খুঁজতে বলেছি যেগুলোর শেষ হয় **.iso** দিয়ে।

**find** কমান্ডের অনেক অনেক অপশন আছে। তারমধ্যে গুরুত্বপূর্ণ কিছু এখানে দিচ্ছি:

টেস্ট	কাজ
-cmin <i>n</i>	<i>n</i> মিনিট আগে বা পরে স্ট্যাটাস মোডিফাইড হয়েছে।
-cnewer <i>file</i>	<i>file</i> এর পরে স্ট্যাটাস বা ডাটা মোডিফাইড হয়েছে এমনসব ফাইল বা ডিরেক্টরি খুঁজবে।
-ctime <i>n</i>	<i>n</i> দিন আগে স্ট্যাটাস মোডিফাইড হয়েছে।
-empty	ফাঁকা ফাইল বা ডিরেক্টরি।
-group <i>name</i>	<i>name</i> নামক গ্রুপের অন্তর্গত।
-iname <i>pattern</i>	নির্দিষ্ট প্যাটার্নের ফাইল বা ডিরেক্টরি খুঁজবে তবে কেস ইনসেনসিটিভ।
-inum <i>n</i>	<i>n</i> ইনোড নম্বরওয়ালা সব ফাইল খুঁজে বের করবে।
-mmin <i>n</i>	<i>n</i> মিনিট আগে বা পরে ডাটা মোডিফাইড হয়েছে।
-mtime <i>n</i>	<i>n</i> দিন আগে ডাটা মোডিফাইড হয়েছে।
-name <i>pattern</i>	নির্দিষ্ট প্যাটার্নের ফাইল বা ডিরেক্টরি খুঁজবে তবে কেস সেনসিটিভ।
-newer <i>file</i>	<i>file</i> এর পরে ডাটা মোডিফাইড হয়েছে এমনসব ফাইল বা ডিরেক্টরি খুঁজবে।
-nouser	এমনসব ফাইল বা ডিরেক্টরি যা কোনো ইউজারের না।
-nogroup	এমনসব ফাইল বা ডিরেক্টরি যা কোনো গ্রুপের না।
-perm <i>mode</i>	পারমিশন মোড অনুযায়ী খুঁজবে।
-samefile <i>name</i>	<i>n</i> ইনোড নম্বরওয়ালা সব ফাইল খুঁজে বের করবে।
-user <i>name</i>	<i>name</i> নামক ইউজারনেমের ফাইল ও ডিরেক্টরি খুঁজবে।

## অপারেটর (operator)

আগের [লেসনে](#) আমরা টেস্টের ব্যবহার দেখেছি। কিন্তু আমরা যদি আরো জটিল টেস্ট চালাতে চাই? মনে করুন শুধু সেই iso ইমেজগুলো খুঁজবো যেগুলো ৩০০ মেগাবাইটের চেয়ে বড় এবং ৮০০ মেগাবাইটের চেয়ে ছোট। তাহলে আমরা কী করব? আমরা কাজটি করতে পারি এভাবে:

```
me@howtocode-pc:~$ find ~ -type f -name "*.iso" \( -size +300M \) -and \( -size -700M \)
/home/nishadsingha/lubuntu-14.04.1-desktop-i386.iso
/home/nishadsingha/archbang-150328-i686.iso
/home/nishadsingha/Porteus-KDE4-v3.1-i486.iso
/home/nishadsingha/debian-7.7.0-amd64-CD-1.iso
```

আমাদের এই কমান্ডে `\( -size +300M \) -and \( -size -700M \)` অংশটুকু নতুন। এখানে আমরা `-size +300M` এবং `-size -700M` এক্সপ্রেশনদুটিকে প্রথমে ব্রাকেটে আবদ্ধ করেছি। যেহেতু শেলের কাছে ব্রাকেটের বিশেষ অর্থ আছে তাই তার সামনে `'\'` ব্যবহার করতে হয়েছে। তারপর এক্সপ্রেশনদুটিকে `-and` অপারেটর দিয়ে যুক্ত করেছি। ফলে `find` সেসব ফাইলই খুঁজবে যেগুলো নামের শেষে `.iso` আছে এবং সেগুলো ৩০০এমবি থেকে বড় এবং ৭০০এমবি থেকে ছোট। অন্যান্য অপারেটরগুলি হলো:

অপারেটর	কাজ
-and	অপারেটরের উভয় পাশের এক্সপ্রেশনের শর্তপূরণ করে এমন ফাইল দেখাবে।
-or	অপারেটরের উভয় পাশের এক্সপ্রেশনের যে কোনো একটি শর্তপূরণ করে এমন ফাইল দেখাবে।
-not	বীপরীত শর্ত তৈরী করবে। যেমন <code>-not -perm 0600</code> সেইসব ফাইলকে দেখাবে যাদের পারমিশন 0600 নয়।
()	টেক্সট এক্সপ্রেশনকে আবদ্ধ করতে ব্যবহৃত হয়।

## একশন (Action)

find দিয়ে কোনো কিছু খুঁজে বের করার পর তার ওপর সরাসরি কোনো কমান্ড এপ্লাই করা যায়। এমনকি এমনকিছু একশন ডিফল্টভাবে দেওয়াও আছে:

একশন	কাজ
-delete	ফাইলগুলো ডিলিট করবে। ব্যবহারে বিশেষ সতর্কতা নিশ্চিত করুন।
-ls	ফাইলগুলো <code>ls -dils</code> কমান্ডের মত প্রিন্ট করবে।
-print	স্ট্যান্ডার্ড আউটপুটে প্রিন্ট করবে। এটি ডিফল্টভাবে কাজ করে।
-quit	কোনো মিল পেলে সার্চ বন্ধ করবে।

যদি আমরা হোমফোল্ডারের সকল .BAK একস্টেনশনওয়ালা ব্যাকআপ ফাইলগুলো ডিলিট করতে চাই তাহলে লিখতে হবে:

```
me@howtocode-pc:~$ find ~ -type f -name '*.BAK' -delete
```

শুধু `find` কমান্ডের নিজস্ব একশন না, আমরাও একশন যোগ করতে পারি এভাবে:

```
me@howtocode-pc:~$ type f -name "*.iso" \( -size +300M \) -and \( -size -700M \) -exec ls
-rw-rw-r-- 1 nishadsingha nishadsingha 729808896 Feb  8 21:31 /home/nishadsingha/lubuntu-
-rw-rw-r-- 1 nishadsingha nishadsingha 411041792 Mar 29 15:39 /home/nishadsingha/archbang
-rw-rw-r-- 1 nishadsingha nishadsingha 478935040 Mar  9 13:40 /home/nishadsingha/Porteus-
-rw-rw-r-- 1 nishadsingha nishadsingha 665845760 Feb  8 21:25 /home/nishadsingha/debian-7
```

আমরা কমান্ডের শেষে `-exec ls -l '{}'` ব্যবহার করেছি। আসুন এর অংশগুলো দেখি:

- **-exec**: এটির মাধ্যমে আমরা কমান্ড এপ্লাই করবো। ইটারএকটিভলি কাজ করতে `-ok` ব্যবহার করতে হবে।
- **ls -l**: আমরা যে কমান্ডটি সার্চ রেজাল্টের উপর এপ্লাই করতে চেয়েছি।
- **'{}'**: এটি দিয়ে কারেন্ট পাথনেম বোঝায়।
- **','**: কমান্ডের সমাপ্তি বোঝায়। এক্ষেত্রে প্রত্যেক সার্চ রেজাল্টের জন্য একবার করে `ls -l` কমান্ডটি দিবে। তবে আমরা যদি `+` ব্যবহার করি আগে সমস্ত সার্চরেজাল্ট থেকে একটি লিস্ট তৈরী করবে, তার ওপর কমান্ডটি এক্সিকিউট করবে।

## xargs

একশন যা করে তা আমরা `xargs` দিয়ে আরো সহজে করতে পারি। `xargs` এর কাজ হল স্ট্যান্ডার্ড আউটপুট থেকে তথ্য নিয়ে অন্য কমান্ডের জন্য ইনপুট তৈরী করা। অনেকসময় ফাইলের নামের ভিতরেই স্পেস থাকে। সেক্ষেত্রে আমরা `find` এর সাথে `-print0` এবং `xargs` এর সাথে `--null` অপশন ব্যবহার করবো। সুতরাং পূর্ববর্তী

কমান্ডটি আমরা এভাবে লিখতে পারি:

```
me@howtocode-pc:~$ find ~ -type f -name "*.iso" \( -size +300M \) -and \( -size -700M \)
-rw-rw-r-- 1 nishadsingha nishadsingha 411041792 Mar 29 15:39 /home/nishadsingha/archbang
-rw-rw-r-- 1 nishadsingha nishadsingha 665845760 Feb  8 21:25 /home/nishadsingha/debian-7
-rw-rw-r-- 1 nishadsingha nishadsingha 729808896 Feb  8 21:31 /home/nishadsingha/lubuntu-
-rw-rw-r-- 1 nishadsingha nishadsingha 478935040 Mar  9 13:40 /home/nishadsingha/Porteus-
```

## অপশন (Option)

আমরা টেস্ট, অপারেশন ও একশন সম্পর্কে জেনেছি। এগুলো সবই অপশন। এর বাইরেও কিছু অপশন ঋে। জেনে নেয়া যাক:

অপশন	কাজ
-depth	যখন <code>find</code> দিয়ে আমরা কোনো কাজ করি তখন কোনো ডিরেক্টরিতে কিছু করার আগে তার ফাইলগুলোতে করবে। <code>-delete</code> একশনে এটি ডিফল্টভাবে এপ্লাই হয়।
-maxdepth <i>levels</i>	ডিরেক্টরি ট্রির কত গভীর পর্যন্ত বিবেচনায় আনবে সেটি ঠিক করে দেওয়া যায় এই অপশন দিয়ে।
-mindepth <i>levels</i>	ডিরেক্টরি ট্রির সর্বনিম্ন কত গভীরতায় কাজ করবে সেটি ঠিক করে দিতে এই অপশনটি ব্যবহৃত হয়।
-mount	মাউন্টেড ড্রাইভের ডিরেক্টরিগুলো আমলে নেবে না।
-noleaf	সিডি/ডিভিডি বা উইন্ডোজের ফাইলসিস্টেমে খুঁজতে এই অপশন দিতে হয় যেন ইউনিক্স এর মত ফাইলসিস্টেমের মত কাজ না করে।



## অনুশীলন

আমরা এবার `find` কমান্ডের যেসব ব্যবহার দেখলাম, আসুন একবার ঝালিয়ে নেওয়া যাক।

প্রথমেই আমরা হোম ডিরেক্টরিতে `playground` নামে একটি ফোল্ডার করবো। যার মধ্যে `dir-001` থেকে শুরু করে `dir-100` পর্যন্ত মোট ১০০ টি ফোল্ডার থাকবে এবং প্রতিটি ফোল্ডারে `file-A` থেকে `file-Z` পর্যন্ত ৩৬ করে ফাইল থাকবে:

```
me@howtocode-pc:~$ mkdir -p playground/dir-{001..100}
me@howtocode-pc:~$ touch playground/dir-{001..100}/file-{A..Z}
```

এখন আমরা সেইসব ফাইল বের করবো খুঁজে যেগুলোর নাম 'file-A':

```
me@howtocode-pc:~$: find playground -type f -name 'file-A'
```

আমরা একটি দীর্ঘ লিস্ট দেখতে পাবো। মোট ফাইলের সংখ্যা জেনে নেবো এভাবে:

```
me@howtocode-pc:~$ find playground -type f -name 'file-A' | wc -l
100
```

এবার আমরা মোডিফিকেশনের সময় অনুযায়ী ফাইল খুঁজবো। এর জন্য আমাদের একটি ফাইল লাগবে যার সাথে আমরা তুলনা করতে পারি। সেটি তৈরী করবো এভাবে:

```
me@howtocode-pc:~$ touch playground/timestamp
me@howtocode-pc:~$ stat playground/timestamp
  File: 'playground/timestamp'
  Size: 0          Blocks: 0          IO Block: 4096   regular empty file
Device: 802h/2050d Inode: 8306121     Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/nishadsingha)   Gid: ( 1000/nishadsingha)
Access: 2015-04-03 15:39:54.671495249 +0600
Modify: 2015-04-03 15:39:54.671495249 +0600
Change: 2015-04-03 15:39:54.671495249 +0600
 Birth: -
```

আমরা `touch playground/timestamp` কমান্ড দিয়ে আমাদের রেফারেন্স ফাইলটি তৈরী করেছি। এবং `stat playground/timestamp` কমান্ড দিয়ে তার সম্পর্কে তথ্যগুলো দেখলাম। আমরা মোডিফিকেশনের সময় দেখতে পাচ্ছি। এবার আমরা কিছু ফাইলকে `touch` কমান্ডের মাধ্যমে নতুন মোডিফিকেশনের সময় দোবো:

```
me@howtocode-pc:~$ find playground -type f -name 'file-B' -exec touch '{}' ';' 
```

আমরা `file-B` নামের সকল ফাইলকে আপডেট করেছি। এবার সেইসব ফাইল খুঁজবো যেগুলো `timestamp` ফাইলটির থেকে মোডিফিকেশনের সময় অনুযায়ী নতুন:

```
me@howtocode-pc:~$ find playground -type f -newer playground/timestamp
```

স্বাভাবিকভাবেই file-B নামের সকল ফাইলই এই লিস্টে থাকবে।

এবার আমরা জটিল একটা কাজ করে শেষ করবো। আমরা দেখবো কোন ফাইলগুলোর পারমিশন **0600** না এবং কোন ডিরেক্টরিগুলোর **0700** না এবং তাদের ওই পারমিশন দেবো:

```
me@howtocode-pc:~$ find playground \( -type f -not -perm 0600 -exec chmod 0600 '{}' ';' \
```



## অধ্যায় - ছয়

# আর্কাইভ ও ব্যাকআপ

তথ্যপ্রযুক্তির যুগের প্রধান ধারক ও বাহক কম্পিউটার, বিভিন্নরকমের কম্পিউটার। আপনার ডেস্কটপ, বৃহৎ সার্ভার, ক্ষুদ্র স্মার্টফোনে অসংখ্য তথ্য। তথ্যের কথা আসলেই আসে সংরক্ষণের কথা, ব্যাকআপ রাখার কথা। তথ্য রাখার জায়গা সীমিত, কাজেই আসে কম্প্রেশনের কথা। এই অধ্যায়ে আর আর্কাইভ করা, কম্প্রেস করা এবং ব্যাকআপ রাখার পদ্ধতিগুলো নিয়ে কথা বলবো।

- **ডাটা কম্প্রেশন:** ডাটা কম্প্রেশন সম্পর্কিত ধারণা ও `gzip` ও `bzip2` এর ব্যবহার।
- **ডাটা আর্কাইভিং:** ডাটা আর্কাইভিং সম্পর্কিত ধারণা এবং `tar` ও `zip` এর ব্যবহার।
- **dtrx:** `dtrx` এর ব্যবহার।
- **সিনক্রোনাইজেশন:** `rsync` এর ব্যবহার।

## ডাটা কম্প্রেশন

ডাটা কম্প্রেশনের মাধ্যমে আমরা বেশি ডাটা কম জায়গায় রাখতে পারি। বাস্তব জীবনে এর ব্যবহার প্রচুর। এমন অনেক কিছুই আছে যা আমরা কম্প্রেসড অবস্থাতেই ব্যবহার করি। যেমন এমপিথ্রি মিউজিক ফাইল বা জেপিইজি ইমেজ।

ডাটা কম্প্রেশন টুলসগুলো ডাটা কম্প্রেসড করতে বিশেষ গাণিতিক পদ্ধতি ব্যবহার করে যাকে কম্প্রেশন এ্যালগরিদম বলে। খুব সহজ থেকে খুব কঠিন, বিভিন্নরকমের এ্যালগরিদম আছে। যেমন মনে করুন, একটা টেক্সট ফাইলে আপনি শুধু 'A' লিখলেন একহাজারবার। ফাইলের মূল ডাটার পরিমাণ হবে ১ হাজার বাইট। এখন আপনি একটি কম্প্রেশন মেথড তৈরী করলেন। সেটা দিয়ে কম্প্রেস করার সময় সেটি লিখলো, "একহাজারটা A". এটুকু লিখতে কিন্তু আপনার একহাজার বাইট লাগছে না। আনকম্প্রেস করার সময় আনকম্প্রেসিঙ টুলটি আবার একহাজারটি 'A' বসিয়ে দেবে। এটি খুব সাধারণ পদ্ধতি কিন্তু আসলে এভাবেও কম্প্রেশন করা হয়। বাস্তবে আরো চমৎকার সব এ্যালগরিদম ব্যবহার করা হয়।

কম্প্রেশন মেথড কীভাবে কাজ করে এটি না জানলেও ক্ষতি ছিল না। পরে মাথা না ঘামালেও চলবে। এবার আমরা দুটি কম্প্রেশন টুল এর ব্যবহার দেখি:

### gzip ও gunzip

gzip ও gunzip যথাক্রমে ফাইল কম্প্রেস ও আনকম্প্রেস করে। আসুন এদের ব্যবহার করা যাক:

```
me@howtocode-pc:~$ ls -l /etc > etc-ls.txt
me@howtocode-pc:~$ ls -l etc-ls*
-rw-rw-r-- 1 me me 15820 Apr  4 19:30 etc-ls.txt
me@howtocode-pc:~$ gzip etc-ls.txt
me@howtocode-pc:~$ ls -l etc-ls*
-rw-rw-r-- 1 me me 3265 Apr  4 19:30 etc-ls.txt.gz
me@howtocode-pc:~$ gunzip etc-ls.txt.gz
me@howtocode-pc:~$ ls -l etc-ls*
-rw-rw-r-- 1 me me 15820 Apr  4 19:30 etc-ls.txt
```

আমরা প্রথমে `ls -l /etc > etc-ls.txt` কমান্ড থেকে /etc ডিরেক্টরির কন্টেন্ট লিস্ট দিয়ে একটি ফাইল বানিয়েছি `etc-ls.txt` নামে। এবার আমরা `ls -l etc-ls*` দিয়েছি এবং দেখেছি ওই ফাইলটির সাইজ 15820 বাইট। এবার `gzip etc-ls.txt` কমান্ড দিয়ে আমরা ফাইলটি কম্প্রেস করেছি। আবারও `ls -l etc-ls*` কমান্ড দিয়ে এবার আমরা দুটো জিনিস লক্ষ্য করেছি। প্রথমত, `etc-ls.txt` ফাইলটি নেই। বরং `etc-ls.txt.gz` তৈরী হয়েছে। এবং এর সাইজ অনেক কম, 3265 বাইট। এবার আমরা `gunzip etc-ls.txt.gz` কমান্ড দিয়ে আনকম্প্রেস করেছি। এবারও `ls -l etc-ls*` কমান্ড দিয়ে আমরা দেখলাম কম্প্রেসড ফাইলটি আনকম্প্রেসড হওয়া `etc-ls.txt` দ্বারা প্রতিস্থাপিত হয়েছে এবং এর সাইজ আগের অবস্থাতেই ফিরে এসেছে।

এবার দেখে নেয়া যাক **gzip** ও **gunzip** এর কিছু অপশন:

অপশন	কাজ
-c	আউটপুট স্ট্যান্ডার্ড আউটপুটে পাঠাবে। যেটাকে আমরা কোনো ফাইলে রিডিবেইট করতে পারি। এতে সুবিধা হল অরিজিনাল ফাইলটি মুছবে না।
-d	ডিকম্প্রেশন মোড। অর্থাৎ gzip gunzip এর মত কাজ করবে।
-f	ইতমধ্যে একইনামে কম্প্রেসড ফাইল থাকলেও নতুন করে কম্প্রেস করবে।
-h	হেল্প দেখাবে।
-l	কম্প্রেসড ফাইলের পরে ব্যবহার করতে হয়। ফাইলটি সম্পর্কে বিভিন্ন তথ্য দেখায়।
-r	রিকার্সিভ মোড।
-t	কম্প্রেসড ফাইল ঠিকঠাক আছে কিনা পরীক্ষা করবে।
-v	ভারবস মোড। বিভিন্ন তথ্য দেখাবে কাজ করার সময়।
-number	নাম্বরের জায়গা 1 থেকে নয়ের মধ্যে নাম্বার দেওয়া যাবে। 1 বা 9 বা --fast দিলে সবচেয়ে দ্রুত কম্প্রেসশন করবে কিন্তু কম্প্রেসড হবে কম। অপরপক্ষে 9 বা --best দিলে সবচেয়ে আস্তে কিন্তু বেশি কম্প্রেসড করবে।

এবার আমরা নীচের কমান্ডটি দেখি:

```
me@howtocode-pc:~$ ls -l /etc | gzip -v > foo.txt.gz
79.5%
```

এখানে আমরা `ls -l /etc` এর ফলাফল পাইপ দিয়ে সরাসরি `gzip -v` কমান্ডে পরিচালিত করেছি এবং তার আউটপুটকে `foo.txt.gz` তে রিডিবেইট করেছি। আমরা এবার এই ফাইলটি টেস্ট করতে পারি এভাবে:

```
me@howtocode-pc:~$ gzip -tv foo.txt.gz
foo.txt.gz: OK
```

আবার আমরা চাইলে আনকম্প্রেস না করেই তথ্য পড়তে পারি এভাবে:

```
me@howtocode-pc:~$ zless foo.txt.gz
```

`zless` কম্প্রেসড ফাইলের ওপর `less` কমান্ডের মত কাজ করে।

## bzip2 ও bunzip2

`bzip2` ও `bunzip2` কমান্ডটি একদম `gzip` ও `gunzip` এর মতই ব্যবহার করতে হয়। তবে এটি তুলনামূলক ধীর কিন্তু আরো উন্নত কম্প্রেশন এ্যালগরিদম ব্যবহার করে। `gzip` ও `gunzip` এর প্রথম উদাহরণটি আমরা

`bzip2` ও `bunzip2` এর জন্য দেখাবো:

```
me@howtocode-pc:~$ ls -l /etc > etc-ls.txt
me@howtocode-pc:~$ ls -l etc-ls.txt
-rw-rw-r-- 1 me me 15820 Apr  4 21:17 etc-ls.txt
me@howtocode-pc:~$ bzip2 etc-ls.txt
me@howtocode-pc:~$ ls -l etc-ls*
-rw-rw-r-- 1 me me 2860 Apr  4 21:17 etc-ls.txt.bz2
me@howtocode-pc:~$ bunzip2 etc-ls.txt.bz2
me@howtocode-pc:~$ ls -l etc-ls*
-rw-rw-r-- 1 me me 15820 Apr  4 21:17 etc-ls.txt
```

আমরা একটা জিনিস এবার লক্ষ্য করছি তা হল এবার কম্প্রেসড ফাইলের এক্সটেনশন bz2, gz নয়। এবং আকারেও ছোট।

# ডাটা আর্কাইভিং

কম্প্রসিং এর সাথে আরো একটি গুরুত্বপূর্ণ কাজ হল আর্কাইভিং (archiving) করা। এর মাধ্যমে অনেক ফাইল মিলে একটি বড় ফাইল তৈরী করা হয়। সিস্টেম ব্যাকআপে এটি বহুল ব্যবহৃত একটি উপায়। এবার আমরা দুটি আর্কাইভিং টুলস সম্পর্কে জানবো:

## tar

`tar` ইউনিক্সসদৃশ সিস্টেমের জগতে একটি ঐতিহ্যবাহী আর্কাইভিং টুল। এর পুরো নাম, **tape archive**। অর্থাৎ যখন ম্যাগনেটিক টেপে ব্যাকআপ রাখা হত তখন এর উৎপত্তি। এখনো এটি কার্যকর পদ্ধতি। আমরা তাই প্রায়ই `.tar`, `.tgz` বা `.tar.gz` এক্সটেনশনের ফাইল দেখি। প্রথমটি সাধারণ আর্কাইভ আর বাকি দুটি কম্প্রসড আর্কাইভ।

`tar` কমান্ডের কাঠামোটি একটু অন্যরকম:

```
tar [-]mode[options] pathname...
```

এখানে মোড বলে একটা অপশন দিতে হয় এবং তারসাথে অন্য অপশন জুড়ে থাকে। এবং অপশনগুলোর সামনে '-' চিহ্ন না দিলেও হয়। এবার আমরা গুরুত্বপূর্ণ মোডগুলো দেখি:

মোড	কাজ
c	নতুন আর্কাইভ তৈরী করে।
x	আর্কাইভ এক্সট্রাক্ট করে।
r	একটি আর্কাইভে আরো ফাইল যোগ করে।
t	আর্কাইভের ফাইল ও ডিরেক্টরির লিস্ট করে।

এবার আর্কাইভিং করতে আমরা কিছু ফাইল ও ডিরেক্টরি তৈরী করি:

```
me@howtocode-pc:~$ mkdir -p playground/dir-{001..100}
me@howtocode-pc:~$ touch playground/dir-{001..100}/file-{A..Z}
```

এবার আমরা playground ডিরেক্টরির সব কন্টেন্টসহ playground.tar নামে একটি আর্কাইভ তৈরী করবো এভাবে:

```
me@howtocode-pc:~$ tar -cf playground.tar playground
```

আমরা চাইলে `-cf` কে `cf` ও লিখতে পারতাম। 'f' অপশন দিয়ে আমরা ফাইলনেম দিয়েছি।

আবার আমরা চাইলে নির্দিষ্ট কিছু ফাইল `find` দিয়ে খুঁজে বের করে আর্কাইভ করতে পারতাম এভাবে:

```
me@howtocode-pc:~$ find playground -name 'file-A' -exec tar -rf playground.tar '{}' '+'
```

এখানে আমরা 'file-A' নামের সকল ফাইল নিয়ে আর্কাইভ করেছি।

আমরা `.tgz` যুক্ত `gzip` কম্প্রেসড আর্কাইভ ও `.tbz` যুক্ত `bzip2` কম্প্রেসড আর্কাইভ তৈরী করতে পারি এভাবে:

```
me@howtocode-pc:~$ tar -czf playground.tgz playground
me@howtocode-pc:~$ tar -cjf playground.tbz playground
```

এখানে 'z' ও 'j' অপশন যথাক্রমে `gzip` ও `bzip2` নির্দেশ করে। সেই অনুসারে আমরা আর্কাইভটিরও নাম বদলেছি।

এবার আমরা যদি চাই সকল `file-A` নামের ফাইল দিয়ে একটি `bzip2` কম্প্রেসড আর্কাইভ করবো তবে লিখতে পারি:

```
me@howtocode-pc:~$ find playground -name 'file-A' | tar -cjf playground.tbz -T -
```

এখানে দুটি বিষয় লক্ষ্যণীয়। প্রথমত আমরা `find` কমান্ডের ফলাফল পাইপ দিয়ে `tar` এ প্রবাহিত করেছি। দ্বিতীয়ত, `tar` কমান্ডের শেষে `-T -` ব্যবহার করেছি। `-` দিয়ে স্ট্যান্ডার্ড ইনপুট ও আউটপুট দুটোই বোঝানো যায় প্রয়োজনমত। এখানে `-T` অপশন দিয়ে আমরা বলেছি কমান্ড থেকে সরাসরি ফাইলের নাম না নিয়ে স্ট্যান্ডার্ড ইনপুট থেকে নিতে। স্ট্যান্ডার্ড ইনপুট বোঝাতে `-` এবং স্ট্যান্ডার্ড ইনপুটে আমরা `find` কমান্ড চালিত করেছিলাম। অর্থাৎ এটি ইনপুট নেবে `find` কমান্ড থেকে।

আমরা `playground.tar` আর্কাইভটির ফাইললিস্ট দেখতে পারি এভাবে:

```
me@howtocode-pc:~$ tar -tf playground.tar
playground/
playground/dir-051/
playground/dir-051/file-B
playground/dir-051/file-E
playground/dir-051/file-I
playground/dir-051/file-H
playground/dir-051/file-W
...
```

আরো বিস্তারিত দেখতে পারি এভাবে:



```
me@howtocode-pc:~$ tar -tvf playground.tar
drwxrwxr-x me/me 0 2015-04-03 15:39 playground/
drwxrwxr-x me/me 0 2015-04-03 14:52 playground/dir-051/
-rw-rw-r-- me/me 0 2015-04-03 15:44 playground/dir-051/file-B
-rw-rw-r-- me/me 0 2015-04-03 14:52 playground/dir-051/file-E
-rw-rw-r-- me/me 0 2015-04-03 14:52 playground/dir-051/file-I
-rw-rw-r-- me/me 0 2015-04-03 14:52 playground/dir-051/file-H
-rw-rw-r-- me/me 0 2015-04-03 14:52 playground/dir-051/file-W
...
```

আমরা এবার নতুন একটি ডিরেক্টরিতে playground.tar এক্সট্রাক্ট করবো:

```
me@howtocode-pc:~$ mkdir extracted-pg
me@howtocode-pc:~$ cd extracted-pg
me@howtocode-pc:~$ tar -xf ../playground.tar
me@howtocode-pc:~$ ls
playground/
```

আমরা extracted-pg নামে একটি ডিরেক্টরি তৈরী করেছি এবং তার মধ্যে এক্সট্রাক্ট করেছি।

আমরা চাইলে সমস্ত আর্কাইভ এক্সট্রাক্ট না করে নির্দিষ্ট এক বা একাধিক ফাইল এক্সট্রাক্ট করতে পারতাম। যেমন আমরা যদি সব file-A নামের ফাইলগুলো এক্সট্রাক্ট করতে চাই সেটা করতে পারি এভাবে:

```
me@howtocode-pc:~$ tar -xf ../playground.tar --wildcards 'home/me/playground/dir-*/file-A'
```

এখানে যেন আমরা পাথনমে ওয়াইল্ডকার্ড ব্যবহার করতে পারি এজন্য --wildcards অপশনটি ব্যবহার করেছি। এবং শেষে ওয়াইল্ডকার্ড এর মাধ্যমে সকল 'file-A' নামের ফাইলগুলো সিলেক্ট করেছি।

## zip ও unzip

zip একই সাথে কম্প্রেস ও আর্কাইভ দুটোই করে। আমরা আমাদের playground ফোল্ডারটি সব কন্টেন্টসহ zip করতে চাইলে লিখতে পারি:

```
me@howtocode-pc:~$ zip -r playground.zip playground
```

-r অপশন দিয়ে রিকার্সিভ বোঝানো হয়েছে।

unzip করতে আমরা লিখতে পারি:

```
me@howtocode-pc:~$ unzip playground.zip
```



## dtrx

প্রাত্যহিক জীবনে আমরা আর্কাইভ ও কম্প্রেস যতটা না করি তারচেয়েও বেশি করি এক্সট্রাক্ট। প্রত্যেকটি আর্কাইভিং ও কম্প্রেসিং মেথডের জন্য আলাদা আলাদা এক্সট্রাক্টিং কমান্ড মনে রাখা একটা ঝামেলা বটে। এই ঝামেলা থেকে মুক্তি দেবে `dtrx`। সাধারণত এই ছোট টুলটি ইন্সটলড থাকেনা ডিফল্টভাবে। তবে রিপোজিটরিতে অবশ্যই পাবেন। আপনি ইন্সটল করে নেবেন।

`dtrx` এর ব্যবহার চূড়ান্তরকমের সোজা। একটা কমান্ডমাত্র মনে রাখবেন:

```
dtrx -rv archive_file...
```

`-r` অপশনটির জন্য যদি আর্কাইভের মধ্যে আরো আর্কাইভ থাকে সেটিও এক্সট্রাক্ট করবে এবং `-v` এর জন্য এক্সট্রাক্ট করার সময় বিভিন্ন তথ্য দেখাবে।

এই পদ্ধতিতে `playground.tar` আর্কাইভটি এক্সট্রাক্ট করতে চাইলে লিখবেন:

```
dtrx -rv playground.tar
```

কখনো কখনো দেখা যায় একটা আর্কাইভে শুধু একটি মাত্র ফাইল বা ডিরেক্টরি থাকে। সেক্ষেত্রে আপনাকে জিজ্ঞাসা করবে কী করতে হবে। আপনি চাইলে স্বাভাবিকভাবে এক্সট্রাক্ট করতে পারেন। আবার বলতে পারেন যে ভিতরের ফাইল বা ফোল্ডারটি শুধু বাইরে এনে এক্সট্রাক্ট করতে।

# সিনক্রোনাইজেশন

নির্দিষ্ট সময় পর পর প্রয়োজনীয় তথ্য ব্যাকআপ রাখা একটা সাধারণ ব্যাপার বিশেষ করে সার্ভারের ক্ষেত্রে। ব্যাকআপ রাখার জায়গাটি কোনো লোক্যাল স্টোরেজ ডিভাইস হতে পারে(এক্সটার্নাল হার্ডডিস্ক বা পেনড্রাইভ) বা রিমোটলি নেটওয়ার্কে সংযুক্ত কোনো কম্পিউটারও হতে পারে। এধরনের ব্যাকআপ রাখতে সবচেয়ে জনপ্রিয় টুল হল `rsync`। এর মাধ্যমে আপনি রিমোট বা লোকাল স্টোরেজে ডাটা সিনক করতে পারবেন। এর কমান্ড কাঠামোটি এরকম:

```
rsync options source destination
```

`source` ও `destination` নীচের যেকোনো একটি হতে পারে:

- লোকাল ফাইল বা ডিরেক্টরি।
- রিমোট ফাইল বা ডিরেক্টরি যাকে এভাবে চিহ্নিত করা যাবে: `[user@]host:path`
- একটি রিমোট সিনক্রোনাইজেশন সার্ভার। যার URI এরকম হবে: `rsync://[user@]host[:port]/path`

উল্লেখ্য, একইসাথে সোর্স ও ডেস্টিনেশন দুটোই রিমোট হতে পারবে না।

এবার আমরা আগে থেকে তৈরী করা `playground` ডিরেক্টরিটি `playground_mirror` নামের ডিরেক্টরির সাথে সিনক্রোনাইজ করবো এভাবে:

```
me@howtocode-pc:~$ mkdir playground_mirror
me@howtocode-pc:~$ rsync playground playground_mirror
...
...
...
sent 137,585 bytes  received 49,867 bytes  374,904.00 bytes/sec
total size is 0  speedup is 0.00
```

আমরা প্রথমে `mkdir` দিয়ে `playground_mirror` ডিরেক্টরিটি তৈরী করেছি। তারপর `rsync -av playground playground_mirror` কমান্ড দিয়ে সিনক করেছি। `-a` অপশন দিয়ে আমরা archive মোড সিলেক্ট করেছি। যার ফলে ফাইল ও ডিরেক্টরির পার্মিশন ও ওনার ইনফরমেশন পরিবর্তিত হবে না। এবং `-v` দিয়ে verbose টিগার করেছি যেন কাজের সময় তথ্য দেখায়।

`rsync` শুধু সেইসব ফাইলই কপি করে যেগুলো পরিবর্তিত হয়েছে। এজন্য এখন আবার আমরা যদি পূর্ববর্তী কমান্ডটি দিই তাহলে এরকম দেখতে পাবো:

```
me@howtocode-pc:~$ rsync -av playground playground_mirror
sending incremental file list

sent 35,823 bytes  received 125 bytes  71,896.00 bytes/sec
total size is 0  speedup is 0.00
```

আমরা দেখতে পাচ্ছি কোনো ফাইল পাঠায়নি। এবার একটা ফাইলকে `touch` কমান্ডের মাধ্যমে নতুন মোডিফিকেশন ডেট দেবো এবং আবার কমান্ডটি দিয়ে দেখবো:

```
me@howtocode-pc:~$ touch playground/dir-051/file-A
me@howtocode-pc:~$ rsync -av playground playground_mirror
sending incremental file list
playground/dir-051/file-A

sent 35,876 bytes  received 150 bytes  72,052.00 bytes/sec
total size is 0  speedup is 0.00
```

আমরা দেখতে পাচ্ছি নতুন মোডিফিকেশন ডেটের ফাইলটি শুধু কপি করেছে।

কখনো কখনো আমরা চাই মিররে সেইসব ফাইল থাকবে না যা মূল জায়গায় নেই সেক্ষেত্রে `--delete` অপশন যোগ করতে হয়। আমরা প্রথমে playground এর কিছু ফাইলসহ একটি ডিরেক্টরি ডিলিট করে দেবো এবং এর উদাহরণ দেখবো:

```
me@howtocode-pc:~$ rm playground/dir-099
me@howtocode-pc:~$ rsync -av --delete playground playground_mirror
sending incremental file list
deleting playground/dir-099/file-Z
deleting playground/dir-099/file-Y
deleting playground/dir-099/file-X
deleting playground/dir-099/file-W
deleting playground/dir-099/file-V
deleting playground/dir-099/file-U
deleting playground/dir-099/file-T
deleting playground/dir-099/file-S
deleting playground/dir-099/file-R
deleting playground/dir-099/file-Q
deleting playground/dir-099/file-P
deleting playground/dir-099/file-O
deleting playground/dir-099/file-N
deleting playground/dir-099/file-M
deleting playground/dir-099/file-L
deleting playground/dir-099/file-K
deleting playground/dir-099/file-J
deleting playground/dir-099/file-I
deleting playground/dir-099/file-H
deleting playground/dir-099/file-G
deleting playground/dir-099/file-F
deleting playground/dir-099/file-E
deleting playground/dir-099/file-D
deleting playground/dir-099/file-C
deleting playground/dir-099/file-B
deleting playground/dir-099/file-A
deleting playground/dir-099/
playground/

sent 35,480 bytes  received 912 bytes  72,784.00 bytes/sec
total size is 0  speedup is 0.00
```

আমরা দেখতে পেলাম আমাদের ডিলিট করা ডিরেক্টরি ও ফাইলসমূহও মিরর থেকে ডিলিট করে ফেলা হল।

## অধ্যায় - সাত

# আটপোরে টুলস

আমরা এই অধ্যায়ে কিছু রোজকার ব্যবহারের টুলস দেখবো। আমরা ব্যবহার করতে শিখবো `ranger` কমান্ডলাইন ফাইল ম্যানেজার, `mutt` কমান্ডলাইন ইমেইল ক্লায়েন্ট, `cmus` কমান্ডলাইন মিউজিক প্লেয়ার, `finch` কমান্ডলাইন আইএম মেসেজিং এ্যাপ, `elinks` কমান্ডলাইন ওয়েব ব্রাউজার এবং `weechat` কমান্ডলাইন আইআরসি ক্লায়েন্ট।

- **রেঞ্জার(Ranger):** ফাইল ম্যানেজার: রেঞ্জারের ব্যবহার।
- **ম্যাট (Mutt):** ইমেইল ক্লায়েন্ট: ম্যাটের ব্যবহার।
- **সিমিউজ(cmus):** মিউজিক প্লেয়ার: সিমিউজের ব্যবহার।
- **ইলিন্কস (elinks):** ওয়েব ব্রাউজার: ইলিন্কসের ব্যবহার।
- **উইচ্যাট (weechat) :** আইআরসি ক্লায়েন্ট: উইচ্যাটের ব্যবহার।
- **ফিন্চ (finch) :** চ্যাট ক্লায়েন্ট ফিন্চের ব্যবহার।

## বেঞ্জার(Ranger): ফাইল ম্যানেজার

আমরা সাধারণ ফাইল ম্যানেজার গ্রাফিকালি ব্যবহার করি। আমার ব্যক্তিগত মতামত হচ্ছে গ্রাফিকাল ফাইল ম্যানেজার এর চেয়ে কমান্ডলাইন ফাইল ম্যানেজার আরো চমৎকার। `ranger` ব্যবহার করতে হলে আপনাকে এটি ইন্সটল করতে হবে। করার পর আপনাকে এর কনফিগারেশন ফাইলগুলো নিজের হোমে কপি করতে হবে এভাবে:

```
me@howtocode-pc:~$: ranger --copy-config=all
creating: /home/nishadsingha/.config/ranger/rifle.conf
creating: /home/nishadsingha/.config/ranger/commands.py
creating: /home/nishadsingha/.config/ranger/rc.conf
creating: /home/nishadsingha/.config/ranger/scope.sh

Please note that configuration files may change as ranger evolves.
It's completely up to you to keep them up to date.
```

এরপর `ranger` কমান্ড দিলে আপনি নীচের মত কিছু দেখতে পাবেন:

```
me@howtocode-pc:/home/me/archives
linux~    archives          1788    partial
me        Assignment         4       account-plugin-aim_3.8.6-0ubuntu9.1_amd6~
phoenix   Bastion               1       account-plugin-jabber_3.8.6-0ubuntu9.1_a~
Storage   Bijoy_keyboard_for_lin~ 11      account-plugin-salut_3.8.6-0ubuntu9.1_am~
          Bijoy_keyboard_for_lin~ 14      account-plugin-yahoo_3.8.6-0ubuntu9.1_am~
          bin                  12      accountsservice_0.6.35-0ubuntu7.1_amd64.~
          bitlbee-facebook     20      android-tools-adb_4.2.2+git20130218-3ubu~
          bn-project          4       apel_10.8+0.20120427-6_all.deb
          c                  5       app-install-data_14.04.1_all.deb
          ccl-1.10-linuxx86    20      apparmor_2.8.95~2430-0ubuntu5.1_amd64.deb
          clisp              0       apport-gtk_2.14.1-0ubuntu3.6_all.deb
          converted         7       apport-gtk_2.14.1-0ubuntu3.7_all.deb
          Desktop          21      apport_2.14.1-0ubuntu3.6_all.deb
          Development       1       apport_2.14.1-0ubuntu3.7_all.deb
          django            1       apt-transport-https_1.0.1ubuntu2.6_amd64~
          django.howtocode.com.bd 9       apt-utils_1.0.1ubuntu2.6_amd64.deb
          Documents         1       apt_1.0.1ubuntu2.6_amd64.deb
          Downloads        127     aptdaemon-data_1.1.1-1ubuntu5.1_all.deb
          Dropbox          6       aptdaemon_1.1.1-1ubuntu5.1_all.deb
          Dropbox (Old)     0       aria2_1.18.1-1_amd64.deb
          e-books           -> 46     artha_1.0.3-1ubuntu1_amd64.deb
          emacs-ipython-notebook 13      aspell-bn_1%3a0.01.1-1-2_all.deb
          emos              2       attr_1%3a2.4.47-1ubuntu1_amd64.deb
          Essential         4       audacity-data_2.0.5-1ubuntu3.2_all.deb
          extracted-pg      1       audacity_2.0.5-1ubuntu3.2_amd64.deb
drwxr-xr-x 3 me me 1788                8.88G sum, 6.84G free 1/165 Top
```

সবার প্রথমে আমরা এই লাইনটি দেখতে পাচ্ছি:



```
me@howtocode-pc:/home/me/archives
```

প্রথমে me@howtocode-pc ইউজার ও হোস্টনেম তারপর /home/me/archives দিয়ে বর্তমান পথ। আপনি নিজের কম্পিউটারে খেয়াল করুন যে পাথনেমের শেষ অংশটুকু, আমার ক্ষেত্রে archives আলাদা রঙের। আমরা আসলে /home/me -তে আছি। এবং archives এর উপরে আমাদের কার্সর রাখা আছে।

সবার শেষে যে লাইনটি আছে তা এরকম:

```
drwxr-xr-x 3 me me 1788      8.88G sum, 6.84G free 1/165 Top
```

এটির বামপাশের অংশে কার্সরের নীচে রাখা ফাইলটি সম্পর্কিত বিভিন্ন তথ্য যা আমরা ls -l কমান্ড দিয়ে দেখতে পাই। এবং ডানদিকে ডিস্কস্পেস সম্পর্কিত বিভিন্ন তথ্য।

মার্কখানের অংশে আমরা তিনটি কলাম দেখতে পাচ্ছি। মার্কখানের কলামটি হচ্ছে সেই ডিরেক্টরি যেখানে আমরা আছি। তার বামপাশে পূর্ববর্তী ডিরেক্টরি। আর ডান পাশে প্রিভিউ কলাম। প্রিভিউ কলাম একটা দারুণ জিনিস। যখন আপনার কার্সর কোনো ডিরেক্টরিতে থাকে তখন ডিরেক্টরির কন্টেন্ট দেখায়। আবার টেক্সট ফাইলের ওপর হলে তার কন্টেন্ট।

## নেভিগেশন

আপনি এন্টার বা রাইট এ্যারো চেপে পরবর্তী ডিরেক্টরিতে যেতে পারেন। কোনো ফাইল খুলতে হলেও আপনাকে এন্টার চাপতে হবে। লেফট এ্যারো চেপে পূর্ববর্তী ডিরেক্টরিতে যেতে পারবেন। আপ ও ডাউন এ্যারো চেপে উপর নীচে ফাইল ব্রাউজ করতে পারবেন।

gn চেপে আপনি নতুন ট্যাব খুলতে পারবেন এবং TAB ও SHIFT-TAB চেপে পরবর্তী ও পূর্ববর্তী ট্যাবে যেতে পারবেন। gc চেপে ট্যাব বন্ধ করতে পারবেন।

কোনো একটি ডিরেক্টরিতে গিয়ে সেটি বুকমার্ক করতে প্রথমে m চাপুন। তারপর শর্টকাট হিসেবে যে বাটন এ্যাড করতে চান সেটি(যেমন a, b, c যেকোনোকিছু হতে পারে।) চাপুন। এবার বুকমার্কে যেতে ` চেপে বুকমার্ক করা বাটনটি চাপুন।

রেঞ্জার বন্ধ করতে চাইলে চাপুন q ।

## কাট, কপি, পেস্ট, রিনেম ও ডিলিট

কাট বা কপি করতে হলে আপনাকে মার্ক করতে হবে আগে। আপনি SPACE চেপে মার্ক করতে পারেন। v চাপলে যেগুলো এতক্ষণ মার্ক করেছেন সেগুলো আনমার্কড হবে এবং অন্যগুলো মার্কড হবে। v চাপলে ভিজুয়াল মোড চালু হবে। যেখানে কার্সর রেখে ভিজুয়াল মোড চালু করবেন তারপর যতদূর যাবেন মার্ক হতে থাকবে। ESC চাপলে মার্ক করা বন্ধ করবে। যেকোনো ধরনের মার্ক বন্ধ করতে uv চাপুন।

ফাইল মার্ক করা হলে কাট করতে dd ও কপি করতে yy চাপুন। আনকাট ও আনকপি করতে ud ও uy চাপতে পারে। এবার যেখানে পেস্ট করতে চান সেখানে গিয়ে pp চাপুন। ওভাররাইট মোডে পেস্ট করতে চাইলে po চাপতে হবে।

কোনো ফাইল রিনেম করতে চাইলে তার ওপরে কাসর রেখে `i` চাপুন। নীচে মিনিবাফারে আপনাকে নতুন নাম জিজ্ঞাসা করবে।

ডিলিট করতে প্রথমে মার্ক করুন তারপর `:delete` লিখে এন্টার দিন। প্রয়োজনমায়িক আপনার কনফার্মেশন চাইবে।

## কনসোল ও টার্মিনাল

`:` চাপলে রেঞ্জারের কনসোল ওপেন হবে নীচের লাইনে। এখানে কমান্ডের নাম ব্যবহার করে কমান্ড দেওয়া যায়।

একইভাবে `!` চাপলে টার্মিনাল ওপেন হবে যেখানে টার্মিনাল কমান্ড দেওয়া যায়।

## ম্যাট (Mutt): ইমেইল ক্লায়েন্ট

কমান্ডলাইন ইমেইল ক্লায়েন্টগুলোর মধ্যে সবচেয়ে জনপ্রিয় বোধহয় ম্যাট ( `mutt` ) এটি ইন্সটলের পর প্রথম কাজ হলো আপনার হোমে `.muttrc` নামে একটি কনফিগারেশন ফাইল তৈরী করা। এরজন্য প্রথমে আপনি নীচের লেখাটুকু কপি করে `.muttrc` ফাইলে পেস্ট করুন। আমরা এই লেসনে ম্যাট টর সাথে জিমেইল একাউন্ট ব্যবহার করছি।

```
# basic .muttrc for use with Gmail
# Change the following six lines to match your Gmail account details
set imap_user = "username@gmail.com"
set imap_pass = "your_password"
set smtp_url = "smtp://username@smtp.gmail.com:587/"
set smtp_pass = "your_password"
set from = "username@gmail.com"
set realname = "Utsob Roy"
set message_cachedir=~/.mutt_msgcache"
#
# # Change the following line to a different editor you prefer.
set editor = 'your_favorite_editor'
# Basic config
set folder = "imaps://imap.gmail.com:993"
set spoolfile = "+INBOX"
set imap_check_subscribed=yes
set hostname = gmail.com
set mail_check = 120
set timeout = 300
set imap_keepalive = 300
set postponed = "+[GMail]/Drafts"
set header_cache=~/.mutt/cache/headers
set message_cachedir=~/.mutt/cache/bodies
set certificate_file=~/.mutt/certificates
set move = no
set include
set sort = 'threads'
set sort_aux = 'reverse-last-date-received'
set auto_tag = yes
set pager_index_lines = 10
ignore "Authentication-Results:"
ignore "DomainKey-Signature:"
ignore "DKIM-Signature:"
hdr_order Date From To Cc
alternative_order text/plain text/html *
auto_view text/html
bind editor <Tab> complete-query
bind editor ^T complete
bind editor <space> noop
bind compose y send-message
# # Gmail-style keyboard shortcuts
macro index,pager am "<enter-command>unset trash\n <delete-message>" "Gmail archive messa
macro index,pager d "<enter-command>set trash=\"imaps://imap.googlemail.com/[GMail]/Bin\"
macro index,pager gi "<change-folder>=INBOX<enter>" "Go to inbox"
macro index,pager ga "<change-folder>=[Gmail]/All Mail<enter>" "Go to all mail"
macro index,pager gs "<change-folder>=[Gmail]/Sent Mail<enter>" "Go to sent mail"
macro index,pager st "<change-folder>=[Gmail]/Starred<enter>" "Go to starred messages"
macro index,pager gd "<change-folder>=[Gmail]/Drafts<enter>" "Go to drafts"
macro index,pager gl "<change-folder>?" "Go to 'Label'" # will take you to a list of all
```

বুঝতেই পারছেন বিভিন্ন জায়গায় কमेंট দিয়ে সেই অংশের কাজ বোঝানো হয়েছে। আমাদের প্রথম অংশটুকু নিজেদের মত করে নিতে হবে। আমরা এই অংশটুকু এডিট করবো:

```
# basic .muttrc for use with Gmail
# Change the following six lines to match your Gmail account details
set imap_user = "username@gmail.com"
set imap_pass = "your_password"
set smtp_url = "smtp://username@smtp.gmail.com:587/"
set smtp_pass = "your_password"
set from = "username@gmail.com"
set realname = "Utsob Roy"
set message_cachedir="~/mutt_msgcache"
#
# # Change the following line to a different editor you prefer.
set editor = 'your_favorite_editor'
```

এখানে যে জায়গাগুলোয় `username` লেখা সেখানে আপনার জিমেইল ইউজারনেম এবং `your_password` এর জায়গায় আপনার পাসওয়ার্ড দেবেন। `your_favorite_editor` এর জায়গায় আপনার এডিটরের কমান্ড। এটি আপনি ন্যানো ব্যবহার করলে ন্যানো দেবেন। আমার মত ইম্যাকস কমান্ডলাইনে ইউজ করলে দেবেন `emacs -nw` আবার গ্রাফিকাল এডিটর যেমন `gedit` ও দিতে পারেন। ফাইলটি সেভ করে `mutt` কমান্ড দিয়ে মাট চালু করুন। এরকম কিছু দেখতে পারেন:

```
q:Quit    u:Undel    m:Mail  r:Reply    ?:Help
 1 N + Apr 07 Gitter Notifica ( 56K) Unread messages in gitterHQ/gitter
 2 N + Apr 07 Freelancer.com   ( 11K) Come back and start earning today!
 3 N + Apr 07 Freelancer.com   (115K) Latest projects and contests matching your
 4 N + Apr 06 Couple App       ( 25K) Welcome to Couple!
 5 N + Apr 06 Ello              ( 31K) Ello! V2 Update + New Greg Foley T-shirt
 6  + Apr 04 Arafat!           (8.8K) Re: রিসোর্সেস
 7   Mar 04 Md. Sabbir Alam (4.5K) [sh.howtocode.com.bd] Changed the SUMMARY.m
 8 r + Mar 04 Md.Sabbir Alam   ( 63) Re: পুন রিকুয়েস্ট প্রস্ক্রিত
 9   Mar 02 Md. Sabbir Alam (5.9K) [sh.howtocode.com.bd] Added a new Chapter (
10  C Mar 03 Md. Sabbir Alam (4.4K) ↳>
11  T Jan 27 Arafat!           (6.5K) Info
12  + Jan 21 The CloudOn Tea ( 26K) CloudOn Joins Dropbox!
13 r + Jan 19 Arafat!          (2.2K) Sorry
---Mutt: =INBOX [Msgs:13 New:5 Flag:3 1.1M]---(threads/reverse-last-date-received)
```

সবচেয়ে উপরের লাইনে আমরা এমনটা দেখতে পাচ্ছি:

```
q:Quit    u:Undel    m:Mail  r:Reply    ?:Help
```

এই লাইনে আপনি বিভিন্ন সময় দরকারি কমান্ডগুলো দেখতে পারবেন। আর অতিরিক্তগুলো দেখতে ? চাপলেই দেখতে পারবেন।

সবার শেষের লাইনটি মিনিবাফার। ঠিক তার উপরের লাইনে আপনি দেখতে পাবেন কোন ফোল্ডারে আছেন, মেসেজের সংখ্যা ইত্যাদি।

## নেভিগেশন

যেকোনো মেইল ফোল্ডারে মেসেজ ব্রাউজ করতে উপরে ও নীচে এ্যারো কী চেপে যেতে পারবেন। এন্টার চাপলে মেসেজটি পড়তে পারবেন। কোনো মেইলের পরবর্তী অংশে যেতে SPACE ও পূর্ববর্তী অংশে যেতে - চাপবেন। মেসেজটি বন্ধ করতে চাপবেন 'i'। বিভিন্ন ফোল্ডারে যেতে আমরা নীচের শর্টকাট ব্যবহার করতে পারি:

শর্টকাট	ফোল্ডার
gi	ইনবক্স
ga	অল মেইল
gs	সেন্ট মেইল
gt	স্টার্ড মেইল
gd	ড্রাফট

## মেইল করা

নতুন মেইল করতে m চাপুন বা কোনো মেইলের রিপ্লাই করতে চাইল তার ওপর কার্সর রেখে r চাপুন। মিনিবাফারে এরকম আসবে:

To:

অর্থাৎ আপনাকে প্রাপকের মেইল এড্রেস দিতে হবে। তারপর সাবজেক্ট জানতে চেয়ে এরকম আসবে:

Subject:

এরপর আপনার ফেডারিট এডিটরে মেইল লেখার ফাইলটি খুলবে। এখানে আপনি লিখবেন, সেভ করবেন এবং বন্ধ করবেন। আপনি সাথে আরও কোনো ফাইল এটাচ করতে চাইলে a চাপবেন। তখন মিনিবাফারে ফাইল পাথ দেখিয়ে দিতে হবে। সবশেষে 'y' চেপে সেভ করবেন।

## সিমিউজ(cmus): মিউজিক প্লেয়ার

সিমিউজ ব্যবহার করতে হলে আপনাকে ইন্সটল করে নিতে হবে। আপনার ডিস্ট্রিবিউশন অনুযায়ী `cmus` প্যাকেজটি ইন্সটল করে নিন। এরপর `cmus` কমান্ড দিয়ে চালু করলে এরকম কিছু দেখবেন:

Artist / Album	Track	Library
. 00:00 - 1:56:00		
playlist   C		

সিমিউজ এর লাইব্রেরীতে কোনো অডিও না থাকায় এরকম ফাঁকা। আমার সমস্ত অডিও `~/Music/` ফোল্ডারে আছে। আমি এগুলো লাইব্রেরীতে লোড করবো এভাবে:

```
:add ~/Music
```

লোড করার পর সমস্ত গান লাইব্রেরীতে চলে এসেছে:

Artist / Album	Track	Library
অজয় চন্দ্র		
অতল জনের গান		
অদিতি মহাশিন		
অনিবদ্য চট্টোপাধ্যায়		
অনুপম রায়		
অর্ণব		
অপেক্ষিতরু বন্দ্যোপাধ্যায়		
আনমনা আনমনা-হেমন্ত ম...		
আমার জীবনপাত্র-হেমন্ত...		
আমার মন মানে না		
আমি তোমার প্রেমে-দ্রাগর...		
আতিথ্য অন্নাম		
আমিনুদ্দীন ডাগর & মইনু...		
আলাউদ্দীন খাঁ		
আলি আরবর খাঁ		
আলি আজমত, বাহাত ফতেহ আ...		
আলি জাফর		
ইন্দ্রদীপ দাশগুপ্ত		
. 00:00 - 1:56:00		playlist   C

## ভিউ

সিমিউজের মোট সাতটি ভিউ আছে যা যথাক্রমে 1-7 কীগুলো দিয়ে আমরা চুকতে পারি। ভিউগুলো হল:

ভিউ	কী	কাজ
লাইব্রেরী	1	আর্টিস্ট ও অ্যালবাম অনুযায়ী সাজানো লাইব্রেরী।
সর্টেড লাইব্রেরী	2	লাইব্রেরীর কন্টেন্ট ইন্ডেক্সের সুবিধা অনুযায়ী সাধারণভাবে সাজানো।
প্লেলিস্ট	3	বর্তমানে ব্যবহৃত প্লেলিস্ট।
প্লে কিউ	4	অডিও ট্র্যাকের কিউ। এখানের ট্র্যাকগুলো প্লেলিস্ট বা লাইব্রেরীর পরবর্তী ট্র্যাকের আগে বাজে।
ব্রাউজার	5	ফাইল ব্রাউজার। এখান থেকে ট্র্যাক লাইব্রেরী, প্লেলিস্ট বা কিউতে যোগ করা যায়।
ফিল্টার	6	ইন্ডেক্স নির্দেশিত বিভিন্ন ফিল্টার সম্পর্কিত তথ্য এখানে থাকে।
সেটিংস	7	সিমিউজ এর সেটিংস।

আমরা ভিউগুলো নিয়ে কথা বলবো। তার আগে কিছু গুরুত্বপূর্ণ শর্টকাট জেনে নেওয়া যায়।

কী	কমান্ড	কাজ
q	quit -i	ইন্টারএকটিভ কুইট কমান্ড, সিমিউজ বন্ধ করতে।
l	echo {}	ট্র্যাক সম্পর্কে তথ্য দেখাবে।
b	player-next	পরবর্তী ট্র্যাক চালাবে।



c	player-pause	পজ করবে। পজ থাকলে প্লে করবে।
x	player-play	প্লে করবে।
z	player-prev	পূর্ববর্তী ট্র্যাক চালাবে।
v	player-stop	স্টপ করবে।
^L	refresh	স্ক্রীন রিফ্রেশ করবে।
n	search-next	ক্রমানুসারে খুঁজবে।
N	search-prev	বীপরীতক্রমে খুঁজবে।
.	seek +1m	ট্র্যাকে এক মিনিট সামনে যাবে।
l, right	seek +5	ট্র্যাকে পাঁচ সেকেন্ড সামনে যাবে।
,	seek -1m	ট্র্যাকে এক মিনিট পিছনে যাবে।
h, left	seek -5	ট্র্যাকে পাঁচ সেকেন্ড পিছনে যাবে।
m	toggle aaa_mode	aaa_mode টুগল করবে।
r	toggle repeat	প্লেলিস্ট বা অ্যালবাম পুনরাবৃত্তি চালু/বন্ধ করবে।
^R	toggle repeat_current	একটি ট্র্যাকের পুনরাবৃত্তি চালু/বন্ধ করবে।
s	toggle shuffle	শাফল্ চালু/বন্ধ করবে।
!	push shell	শেল চালু করবে।
]	vol +0 +1	রাইট স্পীকারে ১ করে ডলিউম বাড়াবে।
[	vol +1 +0	লেফট স্পীকারে ১ করে ডলিউম বাড়াবে।
+, =	vol +10%	১০% ডলিউম বাড়াবে।
}	vol -0 -1	রাইট স্পীকারে ১ করে ডলিউম কমাবে।
{	vol -1 -0	লেফট স্পীকারে ১ করে ডলিউম কমাবে।
-	vol -10%	১০% ডলিউম বাড়াবে।
enter	win-activate	কার্সরের আইটেমটি একটিভ করবে। যেমন ট্র্যাক হলে প্লে করবে।
E	win-add-Q	কিউ এর শুরুতে ট্র্যাক যোগ করবে।
a	win-add-l	লাইব্রেরীতে ট্র্যাক কপি করবে।
y	win-add-p	প্লেলিস্টে ট্র্যাক যোগ করবে।
e	win-add-q	কিউ এর শেষে ট্র্যাক যোগ করবে।
G, end	win-bottom	লিস্টের একদম শেষে যাবে।
down, j	win-down	নীচে নামবে।
p	win-mv-after	কার্সরের ট্র্যাকটি এক ধাপ নীচে নামাবে।

P	win-mv-before	কার্সরের ট্যাকটি এক ধাপ উপরে ওঠাবে।
^F, page_down	win-page-down	এক পেজ নীচে নামবে।
^B, page_up	win-page-up	এক পেজ উপরে উঠবে।
D, delete	win-remove	ট্যাক রিমুভ করবে।
i	win-sel-cur	কার্সর বর্তমানে চালু ট্যাকে নিয়ে যাবে।
space	win-toggle	ট্যাক মার্ক করবে।
g, home	win-top	সবচেয়ে উপরে যাবে।
k, up	win-up	উপরে উঠবে।

`win` দিয়ে শুরু হওয়া কমান্ডগুলো বাদে বাকি সব কমান্ড যেকোনো ডিউতে বসে কাজ করবে।

## লাইব্রেরী ডিউ

আমরা আগেই জেনেছি লাইব্রেরীতে গান যোগ করতে পারেন এভাবে:

```
:add Music_path
```

আবার আপনি লাইব্রেরী ক্লিয়ার করতে পারেন `:clear` দিয়ে।

বামপাশে আপনি লাইব্রেরীর সব আর্টিস্টদের নাম পাবেন। তাদের নামের ওপর কার্সর নিয়ে `space` চাপলে সেই আর্টিস্টের অ্যালবামগুলো দেখাবে। অ্যালবামের ওপর কার্সর নিলে ডানপাশের অংশে গানগুলো দেখতে পাই। `tab` চেপে আমরা দুই অংশের মধ্যে যাতায়াত করতে পারি।

কোনো আর্টিস্টের নামের ওপর কার্সর রেখে এন্টার চাপলে তার সকল অ্যালবাম ও গান চালু হবে। একইভাবে অ্যালবামের ওপর এন্টার চাপলে শুধু অ্যালবাম এবং গান হলে শুধু গান।

## স্টেড লাইব্রেরী ডিউ

এই ডিউতে সকল গান লিস্ট হিসেবে থাকে। যেকোনোটির উপর গিয়ে এন্টার চেপে শুনতে পারেন।

## ব্রাউজার ডিউ

<sup>5</sup> চেপে আপনি যখন ব্রাউজারে যাবেন তখন আপনি ডিরেক্টরী ট্রী দেখতে পাবেন। এন্টার চেপে ফোল্ডারে ঢুকতে পারবেন। কোনো অডিও ফাইল চালাতে চাইলেও এন্টার চাপতে হবে। প্যারেন্ট ডিরেক্টরীতে যেতে ব্যাকস্পেস চাপতে পারে বা ফোল্ডারের সবচেয়ে উপরে থাকা `../` এ কার্সর নিয়ে এন্টার চাপতেও পারেন। এখান থেকে গান কিউতে ও প্লেলিস্টে নিতে যথাক্রমে `e` ও `y` চাপতে হবে।

## কিউ ডিউ

কিউতে থাকা ট্র্যাকগুলো সবচেয়ে বেশি গুরুত্ব পায়। আর সকলকিছুর আগে এগুলো প্লে হয়। আপনি ব্রাউজার বা লাইব্রেরী ডিউ থেকে `e` চেপে যোগ করতে পারেন কিউতে।

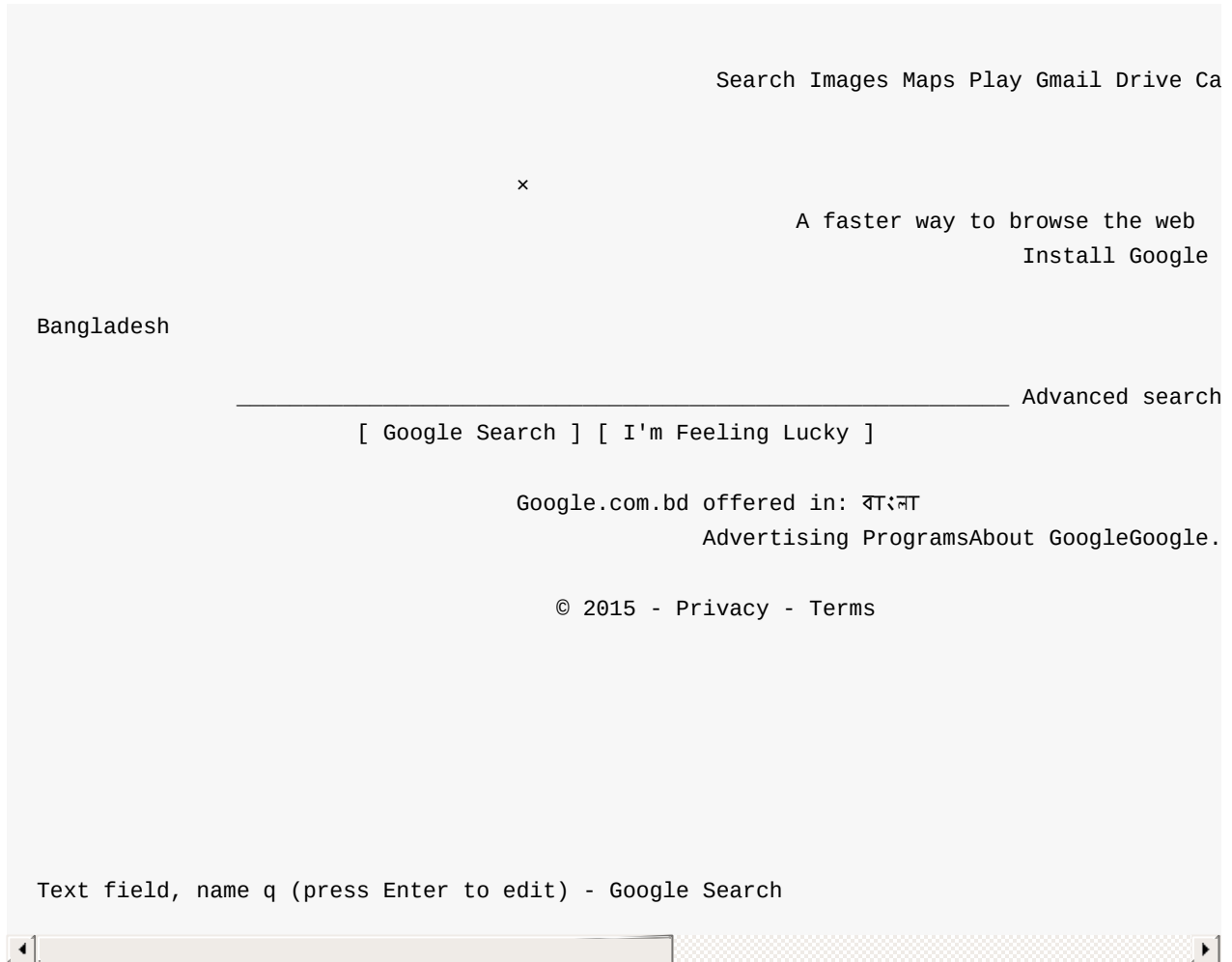
## প্লেলিস্ট ডিউ

আপনি সবচেয়ে বেশি হয়ত এই ডিউটি ব্যবহার করবেন। গান যারা নিয়মিত শোনেন, মুড অনুযায়ী প্লেলিস্ট তৈরী করা তাদের কাছে মিউজিক প্লেয়ারের গুরুত্বপূর্ণ বৈশিষ্ট্য। আমরা যদি নতুন একটা প্লেলিস্ট তৈরী করতে চাই তাহলে প্লেলিস্ট ডিউতে গিয়ে `:clear` কমান্ড দিয়ে প্লেলিস্ট ক্লিয়ার করে নেবো। তারপর ব্রাউজার ডিউতে গিয়ে পছন্দমত ট্র্যাক 'y' চেপে প্লেলিস্টে যোগ করবো। যথেষ্ট পরিমাণে ট্র্যাক যোগ করা হলে `:save path/to/playlist_file.pls` দিয়ে সেভ করবো। এখন আমরা প্লেলিস্টটি ব্যবহার করতে পারি। আমরা একাধিক প্লেলিস্ট এরকম ফাইল হিসেবে সেভ করতে পারি। আমি আমার প্লেলিস্টগুলো একটি ফোল্ডারে রাখি এবং প্রয়োজনমত `:load path/to/playlist_file.pls` দিয়ে লোড করি।

## ইলিন্কস (elinks): ওয়েব ব্রাউজার

কম্পিউটারের সকল সফটওয়্যারের মধ্যে আমাদের সবচেয়ে বড় বন্ধু ওয়েব ব্রাউজার। সকল সমস্যার সমাধানের ইন্টারনেটের জগতে আপনার চলাফেরার মাধ্যম হচ্ছে ওয়েব ব্রাউজার। এই পর্যায়ে আমরা `elinks` এর সাথে পরিচিত হতে যাচ্ছি। যেটা আমার দেখা এপর্যন্ত সবচেয়ে ভালো ওয়েব ব্রাউজার।

ইলিন্কস দিয়ে যদি আমরা `google.com` এ ব্রাউজ করতে চাই তাহলে লিখবো `elinks google.com`। তারপর এরকমকিছু একটা দেখাবে:



এবার দেখে নেওয়া যাক বেসিক কমান্ডগুলো:

কমান্ড	কাজ
g	আমরা নতুন কোনো এড্রেস খুলতে চাইলে g চাপলে এড্রেসবার আসবে।
Home	পেজের সবচেয়ে উপরে উঠবে।
End	পেজের সবচেয়ে নীচে নামবে।
Left	পূর্ববর্তী পেজে যাবে।
Right	কাসরে রাখা লিঙ্কে যাবে।
Enter	একটিভেট করবে। অর্থাৎ হাইপারলিঙ্ক হলে সেই লিঙ্কে যাবে, টেক্সট লেখার জায়গা হলে এন্টার চেপে টেক্সট লিখতে হবে।
Up	আগের হাইপারলিঙ্কে যাবে।
Down	পরের হাইপারলিঙ্কে যাবে।
/	সার্চবক্স আসবে। সার্চ করার পর n চেপে পরবর্তী ও N চেপে পূর্ববর্তী সার্চ রেজাল্টে যেতে পারেন।
[	পেজের বাঁয়ে স্ক্রল করবে।
]	পেজের ডানে স্ক্রল করবে।
d	লিঙ্ক থেকে ডাউনলোড করবে।
ctrl-r	পেজ রিফ্রেশ করবে।
s	বুকমার্ক যাবে।
h	হিস্ট্রিতে যাবে।
o	অপশনসে যাবে।
q	বন্ধ করবে।

তাছাড়াও আপনি `ESC` চাপলে পাবেন একটা মেন্যুবার যেখানে এ্যারো কী ও এন্টারের সাহায্যে আপনি বিভিন্ন কাজ করতে পারবেন।

## ট্যাবড ব্রাউজিং

আধুনিক ব্রাউজারগুলোর মতই `elinks` একাধিক ট্যাব সাপোর্ট করে। এজন্য আপনার এই কমান্ডগুলো কাজে আসবে:

কমান্ড	কাজ
t	নতুন ফাঁকা ট্যাব খুলবে।
T	কার্সরের লিঙ্কটি নতুন ট্যাবে খুলবে।
<	বামদিকের ট্যাবে যাবে।
>	ডানদিকের ট্যাবে যাবে।
Alt-<	ট্যাবটিকে বামে সরাবে।
Alt->	ট্যাবটিকে ডানে সরাবে।
c	ট্যাব বন্ধ করবে।

## উইচ্যাট (weechat) : আইআরসি ক্লায়েন্ট

IRC বা Internet relay chat একটি অতি পুরনো চ্যাট প্রোটোকল। আধুনিক অনেক চ্যাটিং মেথড আসার পরও এটি জনপ্রিয়তা পুরোপুরি হারায়নি। বিশেষকরে প্রোগ্রামাররা এটি এখনো ব্যবহার করেন। এটি কম রিসোর্সে ব্যবহার করা যায় এবং মূলত গ্রুপ চ্যাটের জন্য সুবিধাজনক। অনেকেই অনলাইন মিটিংপ্লেস হিসেবে আইআরসি ব্যবহার করেন। ব্যবহারের জন্য আপনাকে একটি আইআরসি ক্লায়েন্ট এর সাহায্যে সার্ভারের সাথে সংযুক্ত হতে হবে। তারপর আপনি বিভিন্ন রুম (room) এ জয়েন করে চ্যাট করতে পারেন। ইন্টারনেটে বিষয়ভিত্তিক বিভিন্ন রুম পাবেন। প্রয়োজনে নিজেও তৈরী করতে পারেন।

আপনার কম্পিউটারে weechat প্যাকেজটি ইন্সটল করুন এবং তারপর টার্মিনালে লিখুন `weechat`। এমনকি ছু দেখতে পাবেন:

```
WeeChat 1.1.1 (C) 2003-2015 - https://weechat.org/
13:15:35 |  _ _ _ _ _
13:15:35 |  _ |  / / _ _ _ _ _ / _ / _ _ _ / _
13:15:35 |  _ | / / / _ _ \ _ \ / _ _ \ _ \ / _
13:15:35 |  _ | / / / / _ / _ / _ _ / / / / / / _
13:15:35 |  _ / | _ / \ _ \ _ \ _ / / / / \ _ / \ _ /
13:15:35 | WeeChat 1.1.1 [compiled on Jan 27 2015 11:17:07]
13:15:35 | - - - - -
13:15:35 | Bar "input" created
13:15:35 | Bar "title" created
13:15:35 | Bar "status" created
13:15:35 | Bar "nicklist" created
13:15:37 | Plugins loaded: alias, aspell, charset, exec, fifo, guile, irc, logger, lua, p
| relay, ruby, script, tcl, trigger, xfer

[13:15] [1] [core] 1:weechat
```

### সার্ভার সেটআপ

এবার আমাদের একটি সার্ভারে সংযুক্ত হওয়ার পালা। আমরা freenode ব্যবহার করবো। কারন এটি ফ্রী, সবচেয়ে বেশি ব্যবহারকারীও সম্ভবত এখানে এবং #bddevs এর মত বাংলাদেশিদের জন্য অসাধারণ রুমটি এখানে পাবেন। আমাদের সার্ভারের লিস্টে ফ্রীনোড সংযুক্ত করবো এভাবে:

```
/server add freenode chat.freenode.net
```

প্রথমে `/server add` দিয়ে আমরা সার্ভারের লিস্টে এন্ট্রি করার কমান্ড দিচ্ছি। `freenode` হল সার্ভারের নাম। এটি আপনি যেকোনো কিছুর দিতে পারেন। তারপর `chat.freenode.net` হচ্ছে সার্ভারের এড্রেস।

আইআরসিতে সবাই আপনাকে চিনবে আপনার নিকনেম দিয়ে। আমরা কয়েকটা নিকনেম পছন্দ করবো। যেন একটা যদি কেউ ব্যবহার করে তো পরেরটা আমরা নিতে পারি:

```
/set irc.server.freenode.nicks "sh.howtocode,sh_howtocode,sh-howtocode"
```

এখানে আমরা `sh.howtocode`, `sh_howtocode` ও `sh-howtocode` এই তিনটি নিক ব্যবহার করেছি।

এবার আমরা ইউজারনেম এবং রিয়েল নেম সেট করবো এভাবে:

```
/set irc.server.freenode.username "sh.howtocode"  
/set irc.server.freenode.realname "sh howtocode"
```

## কানেকশন

এবার আমরা আমাদের কনফিগার করা freenode সার্ভারে কানেক্ট করতে পারি এভাবে:

```
/connect freenode
```

এবার আমরা #bddevs চ্যানেল/রুমে যুক্ত হবো এভাবে:

```
/join #bddevs
```

জয়েন করার পর আমরা দেখতে পাবো চ্যানেলের স্ক্রীনে তিনটি অংশ। সবচেয়ে বামপাশে সময়, মাঝখানে চ্যাট এবং ডানদিকে ইউজার লিস্ট।

## প্রাইভেট মেসেজ

কোনো ইউজার, মনে করি তার নিকনেম `git_howtocode` হলে তার সাথে প্রাইভেট চ্যাট ওপেন করতে পারি এভাবে:

```
/query git_howtocode
```



## বাফার

প্রত্যেকটি চ্যাট চ্যানেল, সার্ভারগুলি আলাদা আলাদা বাফারে থেলে। আপনি সব বাফারগুলো দেখতে পারেন `/buffer` কমান্ড দিয়ে।

## বন্ধ করা

কোনো বাফার বন্ধ করতে অর্থাৎ চ্যাট চ্যানেল বন্ধ করতে `/close` কমান্ড দিবেন। এবং `weechat` বন্ধ করে `/quit`।

## নিক রেজিস্ট্রেশন

কোনো নিকনেম আগে কেউ রেজিস্টার করে না রাখলে আপনি সেটি রেজিস্টার করে রাখতে পারেন। মনে করুন আপনি এখন `sh-howotocode` নিকটি ব্যবহার করছেন। এটি রেজিস্টার করতে কানেক্টেড অবস্থায় আপনাকে লিখতে হবে:

```
/msg nickserv register my_password my_email_address
```

`my_password` ও `my_email_address` এড্রেসের জায়গায় পছন্দনীয় একটি পাসওয়ার্ড ও একটি সক্রিয় ইমেইল এড্রেস দিতে হবে। তারপর আপনার মেইল এড্রেসে একটি কনফার্মেশন মেইল যাবে। কনফার্ম করার পর নিকটি আপনার জন্য রেজিস্টার্ড হয়ে থাকবে। পরবর্তীতে আপনি অথেনটিকেট করতে পারবেন এভাবে:

```
/msg nickserv identify my_password
```

## অটোমেশন

`weechat` চালু হতেই যদি কোনো সার্ভারে সবসময়ই স্বয়ংক্রিয়ভাবে কানেক্ট করতে চান তাহলে লিখতে পারেন:

```
/set irc.server.server_name.autoconnect on
```

এখানে `server_name` এর জায়গায় `freenode` দিলে আমাদের কনফিগার করা `freenode` সার্ভারে স্বয়ংক্রিয়ভাবে কানেক্ট করবে।

সার্ভারে কানেক্ট করার পর কিছু কাজ আমরা প্রায়ই করতে পারি। যেমন আমাদের নিক আইডেন্টিফাই করতে পারি। কানেক্টের পরই কোনো কমান্ড যদি অটোমেটিকালি ব্যবহার করতে চাই তাহলে তা করতে পারি এভাবে:

```
/set irc.server.freenode.command "/msg nickserv identify my_password"
```

এখানে আমরা `freenode` সার্ভারের জন্য একটি কমান্ড দিচ্ছি যেটা কিনা নিকনেম আইডেন্টিফাই করবে। আমরা একাধিক কমান্ড ; চিহ্ন দিয়ে আলাদা করে একসাথে দিতে পারি।

কানেক্ট করার পর `#bddevs` চ্যানেলে স্বয়ংক্রিয়ভাবে যুক্ত হতে পারি এভাবে:

```
/set irc.server.freenode.autojoin "#bddevs"
```

## অনলাইন হেল্প

সকল কমান্ডের জন্য হেল্প পেতে আপনাকে লিখতে হবে `/help` এবং নির্দিষ্ট কোনো কমান্ড সম্পর্কে জানতে `/help command`। এখানে `command` এর জায়গায় যে কমান্ডটি সম্পর্কে জানতে চান সেটির নাম।

একই ভাবে, সকল অপশনের সম্পর্কে জানতে লিখুন `/set`। আপনি তখন উইচ্যাট ও তার সকল প্লাগিনের অপশনগুলো দেখতে পাবেন। শুধু উইচ্যাটের অপশন দেখতে `/set weechat.*` এবং শুধু আইআরসি প্লাগিনের অপশন দেখতে `/set irc.*` ব্যবহার করুন। নির্দিষ্ট একটি অপশন সম্পর্কে জানতে:

```
/set our.option.name
```

`our.option.name` এর জায়গায় আপনার অপশনটি।

এবার জেনে নেয়া যাক কিছু গুরুত্বপূর্ণ শর্টকাট:

শর্টকাট	কাজ
Alt+left/right Arrow অথবা F5/F6	পূর্ববর্তী বা পরবর্তী বাফারে যাবে।
F7/F8	পূর্ববর্তী বা পরবর্তী উইন্ডোতে যাবে (যদি স্ক্রীন স্প্লিট করা থাকে)।
F9/F10	টাইটেলবারে স্ক্রল করবে।
F11/F12	নিকলিস্টে স্ক্রল করবে।
Tab	শেলের মত লেখা অটোকম্প্লিট করবে।
PgUp/PgDn	বর্তমান বাফারের টেক্সটে স্ক্রল করবে।
Alt+a	যে বাফারে সম্প্রতি কোনো বার্তা এসেছে সেই বাফারে যাবে।

## ফিঞ্চ (finch) : চ্যাট ক্লায়েন্ট

আগের লেসনে আমরা আইআরসি চ্যাটের জন্য ব্যবহার করেছি। এটা অনস্বীকার্য যে আইআরসি থেকে ফেসবুক, গুগল এদের চ্যাটের জনপ্রিয়তা বেশি। এজন্য আমরা অনেকেই পিজিন(pidgin) ব্যবহার করে থাকি গ্রাফিক্যালি। ফিঞ্চকে পিজিনের কমান্ডলাইন ভার্সন মনে করতে পারেন। দুটি সফটওয়্যারই ব্যাকএন্ডে libpurple ব্যবহার করে এবং কনফিগারেশনও শেয়ার করে। `finch` ইন্সটল করুন ও চালু করুন `finch` কমান্ড দিয়ে:

The screenshot shows the 'New Account' dialog box in the Finch application. The dialog is titled 'New Account' and contains the following fields and options:

- Protocol:** A dropdown menu set to 'AIM'.
- Username:** A text input field.
- Password:** A text input field.
- Alias:** A text input field.
- Server:** A text input field set to 'slogin.oscar.aol.com'.
- Port:** A text input field set to '5190'.
- Connection security:** A dropdown menu set to 'Use encryption if ...'.
- Options:**
  - ☐ Remember password
  - ☐ New mail notifications
  - ☒ Use clientLogin
  - ☐ Always use AIM/ICQ proxy server for file transfers and direct IM (slower, but does not reveal your IP address)
  - ☒ Allow multiple simultaneous logins
- Buttons:** 'Cancel' and 'Save' buttons at the bottom.

The dialog is part of a window with tabs for 'New Account', 'Accounts', and 'Buddy List'.

প্রথমবার চালু করার সময় যদি কোনো একাউন্ট না থাকে তাহলে এরকমটা দেখাবে। আমরা এখন একটি গুগল একাউন্ট যোগ করবো:

New Account

list.

Protocol: XMPP v

Username: username

Domain: gmail.com

Resource:

Password: .....

Alias:

☒ Remember password

☐ New mail notifications

☐ Create this account on the server

Connection security Use old-style S... v

☐ Allow plaintext auth over unencrypted streams

Connect port 5223

Connect server talk.google.com

File transfer proxies proxy.eu.jabber.org

BOSH URL

☒ Show Custom Smileys

Cancel Save

New Account

Accounts

Buddy List

আমরা ট্যাব চেপে চেপে ফর্মের ফিল্ডগুলোতে যেতে পারি। আমাদের কনফিগারেশন এরকম ছিল:

- **Protocol:** xmpp
- **Username:** আপনার ইউজারনেম
- **Domain:** gmail.com
- **Password:** আপনার পাসওয়ার্ড
- আমরা Remember Password চেকবক্স একটিভ করেছি স্পেস চেপে যেন বারবার পাসওয়ার্ড দিতে না হয়।
- **Connection security:** Use old style SSL
- **Connect port:** 5223
- **Connect Server:** talk.google.com

এরপর আপনার সকল একাউন্টের ওভারভিউ উইন্ডো আসবে। আপনি Alt-n চেপে বা Buddy List উইন্ডোতে যেতে পারেন। সেখানে অনলাইনে থাকা ইউজারদের পাবেন। যেকোনো ইউজারের নামের ওপর এন্টার চাপলে তারজন্য একটি চ্যাট উইন্ডো খুলবে। আপনি সেখানে চ্যাট করতে পারেন।

এবার দেখে নেওয়া যাক কিছু দরকারি শর্টকাট:

শর্টকাট	কাজ
Alt + a	একশনগুলো দেখাবে। সোজাকথায় পিজিনে আপনি মেন্যুবারে যা যা পাবেন সেগুলো দেখাবে।
Alt + n	পরবর্তী উইন্ডোতে যাবে।
Alt + p	পূর্ববর্তী উইন্ডোতে যাবে।
Alt + w	উইন্ডো লিস্ট দেখাবে যেখান থেকে আপনি যেকোনো উইন্ডোতে যেতে পারবেন।
Alt + c	বর্তমান উইন্ডো বন্ধ করবে।
Alt + q	ফিঞ্চ বন্ধ করবে।
Alt + Tab	পরবর্তী আর্জেন্ট উইন্ডোতে যাবে।
Alt + Shift + Tab	পূর্ববর্তী আর্জেন্ট উইন্ডোতে যাবে।
Ctrl + o অথবা F10	বর্তমান উইন্ডোর মেন্যু দেখাবে (যদি থাকে)।
Ctrl + x অথবা F11	বর্তমান উইন্ডোর কনটেন্ট মেন্যু দেখাবে (যদি থাকে)।

## অধ্যায় - আট

# প্রোগ্রাম কম্পাইলেশন

কম্পাইলেশন অর্থ কোনো প্রোগ্রামের সোর্সকোড থেকে ব্যবহারযোগ্য বাইনারীতে রূপান্তরিত করা। আমরা হরহামেশাই কম্পাইল করি তার কারন এই না যে এটা করার মধ্যে ভাব আছে, বলা যেতে পারে বাধ্য হয়েই। কখনো কখনো কোনো সফটওয়্যার আপনার ডিস্ট্রিবিউশনের রিপোজিটরিতে না থাকতে পারে। অথবা লেটেস্ট ভার্সনটি এখনো আসেনি, কম্পাইল করা ছাড়া উপায় নেই।

প্রত্যেকটি প্রোগ্রাম একইভাবে কম্পাইল করা যায় না। একটুআধটু পার্থক্য থাকে। আসলে একটা কম্পাইলড প্রোগ্রাম আর বাইনারির মাঝের অংশটুকু কমনসেন্স।

এবার আমরা বাঙলা লেখার অন্যতম জনপ্রিয় প্রোগ্রাম অন্ড কীবোর্ডের লিনাক্স ভার্সন আইবাস-অন্ড (ibus-avro) কম্পাইল করবো। কম্পাইল করার জন্য আমি আর্চ লিনাক্স ৬৪বিট ব্যবহার করছি।

## সোর্সকোড সংগ্রহ

কম্পাইল করতে গেলে প্রথমে আপনার সোর্সকোড প্রয়োজন। গুগলে ibus-avro লিখে সার্চ দিলে আপনি এর গিটহাব রিপোজিটরি পাবেন যেখানে সোর্সকোড রাখা আছে। সোর্সকোড সংগ্রহের বিভিন্ন উপায় আছে। আপনি অনেকসময় tar.gz আর্কাইভ হিসেবে পেতে পারেন। তো, গিট থেকে ক্লোন করতে আমাদের git ইন্সটলড থাকতে হবে। রিপোজিটরির বামপাশে একটি বক্সে ক্লোন ইউআরএল পাবেন। এবার আমরা ক্লোন করবো এভাবে:

```
me@howtocode-pc:~$ git clone https://github.com/sarim/ibus-avro.git
Cloning into 'ibus-avro'...
remote: Counting objects: 1061, done.
remote: Total 1061 (delta 0), reused 0 (delta 0), pack-reused 1061
Receiving objects: 100% (1061/1061), 10.55 MiB | 54.00 KiB/s, done.
Resolving deltas: 100% (491/491), done.
Checking connectivity... done.
```

## কনফিগার, কম্পাইল ও ইন্সটল

এবার আমরা ক্লোন করা সোর্সকোডের ফোল্ডারে ঢুকবো এবং সবার আগে README ফাইলটা পড়বো। সাধারণত রিডমি ফাইলে কম্পাইল করার পদ্ধতি বলা থাকে:

```
me@howtocode-pc:~$ cd ibus-avro
me@howtocode-pc:~$ less README.md
```

ইন্সটলেশনের প্রথম ধাপটি এরকম:

1. Open terminal/package manager and install following packages:

```
git
libibus-1.0-0
libibus-1.0-dev
ibus
automake
autoconf
gjs
gir1.2-gjsdbus-1.0
gir1.2-ibus-1.0
```

\_\_For Ubuntu 12.04\_\_

```
sudo apt-get install git ibus libibus-1.0-dev automake autoconf gjs gir1.2-gjsdbu
```

\_\_For other linux distributions\_\_

You'll need all related build tools like `automake`, `autoconf` etc...

and Latest \_\_IBus\_\_ from \_git\_ compiled with \_gobject-introspection\_ support enab

তো আমরা দেখছি যে, অত্র ইন্সটল করতে গেলে আমাদের কিছু ডিপেন্ডেন্সি ইন্সটল করতে হবে। উবুন্টু ১২.০৪ এর জন্য সরাসরি কমান্ডই দেওয়া আছে। আমরা ইতমধ্যে git ইন্সটল করেছি। আইবাসের লেটেস্ট ভার্সনে gir লাইব্রেরী দুটি দেওয়াই থাকে। আমরা তাই libibus, ibus, automake, autoconf gjs ইন্সটল করলেই হবে। আপনাকে gir লাইব্রেরী দুটি ইন্সটল করতে হবে যদি উবুন্টু ব্যবহার করে থাকেন। ডেবিয়ান, আর্চ লিনাক্স এগুলোতে প্রয়োজন হবে না।

তারপর আমরা দ্বিতীয় ধাপে দেখছি:

2. Now give the following commands step-by-step:

```
git clone git://github.com/sarim/ibus-avro.git
cd ibus-avro
aclocal && autoconf && automake --add-misig
./configure --prefix=/usr
sudo make install
```

আমরা ইতমধ্যে সোর্সকোড ক্লোন করেছি ও সেই ডিরেক্টরিতেই আছি। অতএব প্রথম দুইলাইন দ্বিতীয়বার করতে হবে না। তারপরই এই কমান্ডটি দিতে হবে:

```
aclocal && autoconf && automake --add-misig
```

এই কমান্ডটির মাধ্যমে আমরা Makefile তৈরী করবো। তারপর কনফিগার করবো এই কমান্ড দিয়ে:

```
./configure --prefix=/usr
```

`--prefix=/usr` দিয়ে আমরা বলছি যে কম্পাইল করার পর তা `/usr` ডিরেক্টরিতে ইন্সটল করতে।

সবশেষে `sudo make install` দিয়ে আমরা কম্পাইল করে ইন্সটল করলাম।

সম্পূর্ণ কম্পাইলেশনে আসলে আমাদের মাথা খাটানোর কিছু নেই। রিডমি ফাইল পড়লে সহজেই বোঝা যায় কীভাবে কী করতে হবে।



## চতুর্থ খণ্ড

### টেক্সট ম্যানিপুলেশন

এই খণ্ডের বিষয়বস্তু টেক্সট। তার কিছু অংশ আমাদের পরবর্তীতে শেলস্ক্রিপ্টিং এর জন্য প্রয়োজন হবে যেমন রেগুলার এক্সপ্রেসন। তাছাড়াও জানবো টেক্সট প্রসেসিং এর খুঁটিনাটি, জানবো আউটপুট ফরম্যাটিংয়ের উপায় এবং সবশেষে প্রিন্টিং।

- প্রথম অধ্যায় - রেগুলার এক্সপ্রেসন

## অধ্যায় - এক

# বেগুলার এক্সপ্ৰেশন

সহজ কথায় বেগুলার এক্সপ্ৰেশন হচ্ছে টেক্সটের মধ্যে নির্দিষ্ট প্যাটার্ন খুঁজে বের করার একটি উপায়। আমরা ওয়াইল্ডকার্ড ব্যবহার করেছি। খানিকটা সেরকমই। যেসব টুল টেক্সট নিয়ে কাজ করে তারা অধিকাংশই বেগুলার এক্সপ্ৰেশন সাপোর্ট করে। তবে সকল প্রোগ্রাম বা টুল একইভাবে বেগুলার এক্সপ্ৰেশনের সাপোর্ট দেয় না। উনিশ-বিশ পার্থক্য তো থাকেই, বড় পার্থক্যও থাকে। যেমন আমাদের পূর্বব্যবহৃত `grep` POSIX স্ট্যান্ডার্ড মেনে চলে অন্যদিকে পার্ল প্রোগ্রামিং ল্যাঙ্গুয়েজ আরো বেশি ক্ষমতাধর বেগুলার এক্সপ্ৰেশন ব্যবহার করে। আমরা `grep` নিয়েই কাজ করবো।

- **গ্রেপ (grep):** গ্রেপের ব্যবহার।

## গ্রেপ (grep)

`grep` নিয়ে আমরা আগেও কাজ করেছি। `grep` এর পুরো অর্থ "global regular expression print"। বুঝতেই পারছেন। রেগুলার এক্সপ্রেশনই এর বিশেষত্ব। এর কাজই হলো টেক্সট ফাইল থেকে সেইসব লাইন খুঁজে বের করা যেগুলো প্যাটার্নে মিলবে। গ্রেপকে আমরা এ পর্যন্ত পাইপ এর দ্বারা ব্যবহার করেছি। গ্রেপের কমান্ডকাঠামোটি এরকম:

```
grep [options] regex [file...]
```

অর্থাৎ প্রথমে `grep` তারপর অপশনসমূহ, তারপর `regex` অর্থাৎ রেগুলার এক্সপ্রেশন এবং সবশেষে এক বা একাধিক ফাইল।

এবার দেখে নেওয়া যাক গ্রেপে ব্যবহৃত কিছু অপশন:

অপশন	লং অপশন	কাজ
-i	--ignore-case	বড় ও ছোটহাতের লেখার মধ্যে পার্থক্য করবে না। দুটোই গ্রহণ করবে।
-v	--invert-match	যেসব লাইন মিলবে সেগুলো দেখানোর বদলে যেগুলো মিলবে না সেগুলো দেখাবে।
-c	--count	লাইনগুলো না দেখিয়ে মোট কতগুলো লাইনে মিলেছে তার সংখ্যা জানাবে।
-l	--files-with-matches	মিলে যাওয়া লাইন না দেখিয়ে লাইনগুলো কোন ফাইলের তার নাম দেখাবে।
-L	--file-without-match	-l অপশনের মত তবে উল্টো। অর্থাৎ সেই ফাইলগুলোর নাম দেখাবে যেগুলোতে মেলেনি।
-n	--line-number	মিলে যাওয়া লাইনের নম্বর দেখাবে।
-h	--no-filename	একাধিক ফাইলে খোঁজার সময় ফাইলের নাম দেখাবে না।

এবার গ্রেপের শক্তিপরীক্ষার জন্য আমাদের কিছু ফাইল দরকার। অতএব, কিছু ফাইল তৈরী করা যাক:

```
me@howtocode-pc:~$ ls /usr/bin > dirlist-usr-bin.txt
me@howtocode-pc:~$ ls /usr/lib > dirlist-usr-lib.txt
me@howtocode-pc:~$ ls /usr/local > dirlist-local.txt
me@howtocode-pc:~$ ls /usr/share > dirlist-usr-share.txt
me@howtocode-pc:~$ ls dirlist*.txt
dirlist-usr-bin.txt  dirlist-usr-lib.txt  dirlist-usr-local.txt  dirlist-usr-share.txt
```

তো, এবার দেখা যাক আমাদের এই চারটি ফাইলে `bzip` কোথায় কোথায় লেখা আছে:

```
me@howtocode-pc:~$ grep bzip dirlist*.txt
dirlist-usr-bin.txt:bzip2
dirlist-usr-bin.txt:bzip2recover
dirlist-usr-lib.txt:libzip
dirlist-usr-lib.txt:libzip.so
dirlist-usr-lib.txt:libzip.so.2
dirlist-usr-lib.txt:libzip.so.2.1.0
```

আমরা দেখছি প্রত্যেকলাইনে একটি করে ফাইলের নাম এবং সেই ফাইলে bzip এর সাথে মিলে যাওয়া লাইনটি। এবার আমরা যদি চাইতাম শুধু দেখবো কোন কোন ফাইলে bzip আছে সেটা এভাবে করতে পারি:

```
me@howtocode-pc:~$ grep -l bzip dirlist*.txt
dirlist-usr-bin.txt
dirlist-usr-lib.txt
```

এবার শুধু ফাইলের নাম দেখাচ্ছে। এবার আমরা যদি দেখতে চাই কোন কোন ফাইলের মধ্যে bzip নাই, সেটা দেখতে পারি এভাবে:

```
me@howtocode-pc:~$ grep -L bzip dirlist*.txt
dirlist-usr-local.txt
dirlist-usr-share.txt
```

আমরা দেখতে পাচ্ছি কোনো দুটি ফাইলে bzip নাই।

## লিটারাল এবং মেটাক্যারেটর (Literals and Metacharacter)

এপর্যন্ত আমরা খুব সাধারণ রেগুলার এক্সপ্রেশন বা রেজেক্স ব্যবহার করেছি। যার মধ্যে ছিল শুধু লিটারাল ক্যারেটর। লিটারাল হচ্ছে সেইসব ক্যারেটর যেগুলোর বিশেষ কোনো অর্থ নাই রেজেক্সে। যেমন a, b, ই, ড ইত্যাদি। অর্থাৎ, a বলতে a-ই বুঝবে। আমাদের পূর্ববর্তী রেজেক্স ছিল bzip। যেখানে b, z, i ও p এই চারটি লিটারাল ক্যারেটর খুঁজতে বলেছি যাদের মধ্যে আর কোনো ক্যারেটর থাকবে না।

অন্যদিকে আছে কিছু মেটাক্যারেটর। রেজেক্সে মেটাক্যারেটরগুলোর বিশেষ অর্থ আছে। আমরা যেমন ওয়াইল্ডকার্ডের জন্য \* ও ? ব্যবহার করেছি সেরকম। গ্রেপ এই মেটা ক্যারেটরগুলো ব্যবহার করে:

```
^$.[]{}()-?*+|\
```

এগুলো ছাড়া বাকি সব ক্যারেটর লিটারাল।

মনে রাখা প্রয়োজন যে এইসব ক্যারেটরের শেলের কাছেও বিশেষ অর্থ আছে। তাই সম্পূর্ণ রেজেক্সকে " চিহ্ন দিয়ে কোট করে দেওয়া ভালো যেন শেল এগুলোকে এক্সপ্যান্ড করার চেষ্টা না করে।

## এনি ক্যারেটর (Any Character)

আমরা যদি রেজেক্সে ডট(ফুলস্টপ) চিহ্ন ব্যবহার করি তাহলে ডটের স্থানে যেকোনো ক্যারেঞ্জার ম্যাচ করবে। একটা উদাহরণ দেখা যাক:

```
me@howtocode-pc:~$ grep -h '.zip' dirlist*.txt
bunzip2
bzip2
bzip2recover
funzip
gpg-zip
gunzip
gzip
hunzip
hzip
lrzip
lrzip
preunzip
prezip
prezip-bin
unzip
unzipsfx
liblrzip.so
liblrzip.so.0
liblrzip.so.0.0.0
libzip
libzip.so
libzip.so.2
libzip.so.2.1.0
p7zip
```

আমাদের রেজেক্স ছিল `.zip` অর্থাৎ `zip` এর আগে অন্তত একটি যেকোনো ক্যারেঞ্জার থাকতে হবে। এজন্যই, হয়ত আপনি লক্ষ্য করেছেন ফলাফলে `zip` নেই। কারন `zip` এর আগে কোনো ক্যারেঞ্জার নেই।

## এ্যান্কার (Anchor)

রেজেক্সে দুটি এ্যান্কার মেটাক্যারেঞ্জার আছে। `^` (ক্যারেট) এবং `$` (ডলার) চিহ্ন। প্রথমটি রেজেক্সের শুরুতে ও দ্বিতীয়টি শেষে বসে। প্রথমটি শুধু সেইসব ম্যাচ বের করে যেগুলোর শুরুতে রেজেক্সটি আছে। দ্বিতীয়টি খুঁজে বের করে যেগুলোর শেষে থাকে। আসুন, ব্যবহার করে দেখি:

```
me@howtocode-pc:~$ grep -h '^zip' dirlist*.txt
zip
zipcloak
zipcmp
zipgrep
zipinfo
zipmerge
zipnote
zipsplit
ziptorrent
me@howtocode-pc:~$ grep -h 'zip$' dirlist*.txt
funzip
gpg-zip
gunzip
gzip
hunzip
hzip
lrzip
lrzip
preunzip
prezip
unzip
zip
libzip
p7zip
me@howtocode-pc:~$ grep -h '^zip$' dirlist*.txt
zip
```

প্রথমে আমরা সেইসব ম্যাচ খুঁজলাম যার শুরুতে zip আছে এবং পরবর্তীতে খুঁজলাম কোনগুলোর শেষে আছে।  
সবার শেষে দেখলাম কোনগুলোর শুরু এবং শেষে zip আছে। স্বাভাবিকভাবেই zip ছাড়া আর কোনো ম্যাচ ছিল না।