မြန်မာကျောင်းသားများအတွက် အခြေခံ bash command များ

bash or linux command တွေကို ကောင်းကောင်းသုံးတတ်ရင် ကွန်ပြူတာနဲ့ အလုပ်လုပ်တဲ့အခါမှာ ပိုပြီးမြန်ဆန်လာပါလိမ့်မယ်။

ဒီ tutorial က မြန်မာ ကျောင်းသားတွေအားလုံးကို ရည်ရွယ်ပါတယ်။
အထူးသဖြင့် ကွန်ပြူတာတက္ကသိုလ်တွေ၊ အင်ဂျင်နီယာတက္ကသိုလ်တွေက မဟာဘွဲ့၊ ဒေါက်တာဘွဲ့ ယူဖို့ ရည်ရွယ်ထားကြသူတွေ၊ သုတေသနကို လုပ်ချင်ကြတဲ့ ကျောင်းသားတွေက linux command တွေကို ကျွမ်းကျင်စွာ သုံးနိုင်ကြရမယ်။ ဘာ့ကြောင့်လဲ ဆိုတော့ ခင်ဗျားတို့ linux ကို မသုံးလို့ကို မဖြစ်လို့ပါ။

သုတေသနအတွက်၊ လက်ရှိရှိနေပြီးသား သုတေသနပရိုပို့ဇယ်တွေမှာသုံးထားကြတဲ့ ပရိဂရမ်တွေ၊ မို့ငြမ်းလို့ ရနိုင်မဲ့ နမူနာ prototype system တွေ၊ open source အနေနဲ့ ရရှိနိုင်တဲ့ source code အများစုကလည်း linux OS ပေါ် မှာပဲ develop လုပ်ကြတာမို့လို့ပါ။ နောက်ပြီးတော့ linux/Unix command (သို့) command line interface (CLI) ကို အသုံးပြုရခြင်းရဲ့အားသာတဲ့ အချက်တွေကို သိထားသင့်ပါတယ်။

လေ့လာတဲ့အခါ ရရှိမဲ့ နောက်ဆက်တွဲ ဗဟုသုတနဲ့ အကျိုးကျေးဇူးတွေက အများကြီးမို့လို့ပါ။

အဲဒါကြောင့် ပထမဆုံးအဆင့်အနေနဲ့ လက်တွေ့မှာ အသုံးဝင်မဲ့၊ မသိမဖြစ်သိသင့်တဲ့ command တွေကို အသုံးများတဲ့ option တွေကိုပဲ သုံးပြီး ရှင်းပြပါမယ်။ Command တွေကို လက်တွေ့ စမ်းရိုက်၊ မော်နီတာစကရင်မှာ မြင်ရတဲ့ output တွေကို လေ့လာကြည့်ရင်း၊ မြန်မြန်ဆန်ဆန် နားလည်ပြီး၊ ကိုယ်လုပ်ချင်တဲ့ သုတေသနအလုပ်တွေအတွက် စတင်အသုံးပြုနိုင်ဖို့ကို အဓိက ရည်ရွယ်ပါတယ်။

1. ls (list directory contents)

လက်ရှိ ရောက်နေတဲ့ ဖိုလ်ဒါ မှာရှိတဲ့ ဖိုင်တွေ၊ ဖိုလ်ဒါတွေကို ကြည်ချင်ရင်

In [1]:

```
ls
```

```
all-vi-info.txt  data5     folderA                  ls.out   otest.tag
data1            echo      folder-new               my-news  otest.wor
d
data2            fileA     item-list                newfile  screen
data3            fileB     linux-commands.ipynb     news     sorted-na
mes
data4            fmt.out   linux-commands.ipynb.bak  otest    wildcard
```

လက်ရှိရောက်နေတဲ့ ဖိုလ်ဒါအောက်မှာ ရှိတာတွေကို ရိုက်ပြပါလိမ့်မယ်။

အဲဒီ နာမည်တွေက ဖိုင်တွေလည်းဖြစ်နိုင်သလို၊ ဖိုလ်ဒါတွေကော မဖြစ်နိုင်ဘူးလားလို့ မေးရင်၊ Linux OS မှာက ဖိုလ်ဒါဆိုရင် နာမည်နောက်က / (forward slash) လိုက်တယ်။ ပုံမှန်အားဖြင့်က setting မလုပ်ထားရင် / ကိုတွဲမပြဘူး။ ဘာ့ကြောင့်လဲ ဆိုတော့ တကယ်တမ်းက ဖိုင်နဲ့ ဖိုလ်ဒါနဲ့က အရောင်မတူဘူး။ အဲဒါကြောင့် အရောင်ကို ကြည့်ပြီး ဖိုင်လား၊ ဖိုလ်ဒါလားက သိသာလို့ပါ။

ဖိုလ်ဒါ အသစ် တစ်ခုဆောက်ချင်ရင် mkdir ဆိုတဲ့ command ကိုသုံးတယ်။

In [2]:

```
mkdir folder-athit
```

ls command ကိုပြန်ရိုက်ကြည့်မယ်။ ဆောက်လိုက်တဲ့ ဖိုလ်ဒါအသစ် folder-athit ကိုမြင်ရလိမ့်မယ်။

ဖိုလ်ဒါတွေကို / နဲ့တွဲပြစေချင်ရင် ls command ကို option -p ထည့်ပြီး run ကြည့်ပါ။

In [3]:

```
ls -p
```

```
all-vi-info.txt  fileA          linux-commands.ipynb      otest.tag
data1/           fileB          linux-commands.ipynb.bak  otest.word
data2/           fmt.out        ls.out                    screen/
data3/           folderA/       my-news                   sorted-names
data4/           folder-athit/  newfile                   wildcard/
data5/           folder-new/    news
echo             item-list      otest
```

ဖိုလ်ဒါ ဖြစ်တဲ့ folder-athit ရဲ့နောက်မှာ / ပါလာတာကို တွေ့ရလိမ့်မယ်။

ဖိုင်အမျိုးအစားတွေကို အသေးစိတ်ပြပေးစေချင်ရင် -F option ကိုသုံးပြီးကြည့်တယ်။

In [4]:

```
ls -F /usr/bin/r*
```

```
/usr/bin/ranlib@            /usr/bin/rfcomm*
/usr/bin/rasttopnm*         /usr/bin/rgb3toppm*
/usr/bin/rawtopgm*          /usr/bin/rgrep*
/usr/bin/rawtoppm*          /usr/bin/rhythmbox*
/usr/bin/rcc@               /usr/bin/rhythmbox-client*
/usr/bin/rcp@               /usr/bin/rimraf@
/usr/bin/rctest*            /usr/bin/rletopnm*
/usr/bin/readelf@           /usr/bin/rlogin@
/usr/bin/realpath*          /usr/bin/rmid@
/usr/bin/recode-sr-latin*   /usr/bin/rmiregistry@
/usr/bin/recountdiff*       /usr/bin/routef*
/usr/bin/rediff*            /usr/bin/routel*
/usr/bin/remmina*           /usr/bin/rpcgen*
/usr/bin/rename@            /usr/bin/rsh@
/usr/bin/rename.ul*         /usr/bin/rstart*
/usr/bin/rendercheck*       /usr/bin/rstartd*
/usr/bin/renice*            /usr/bin/rsync*
/usr/bin/reset@             /usr/bin/rtstat@
/usr/bin/resize*            /usr/bin/runcon*
/usr/bin/resizecons*        /usr/bin/run-mailcap*
/usr/bin/resizepart*        /usr/bin/run-with-aspell*
/usr/bin/rev*               /usr/bin/rview@
```

-F option နဲ့ /usr/bin/ ဖိုလ်ဒါအောက်မှာရှိတဲ့ r စာလုံးနဲ့စတဲ့ ဖိုင်တွေကို ပြခိုင်းတဲ့ အခါမှာ ဖိုင်နာမည်တွေရဲ့နောက်မှာ @ နဲ့ * အမှတ်အသားတွေပါလာတာကို တွေ့ရပါ လိမ့်မယ်။ * အမှတ်အသားက executable လုပ်လို့ရတဲ့ဖိုင်၊ run လို့ရတဲ့ ဖိုင်ကို ဆိုလိုပြီး၊ @ အမှတ်အသားကတော့ symbolic link အနေနဲ့ ချိတ်ထားတဲ့ ဖိုင်လို့ ဆိုလိုတာပါ။
symbolic link ကိစ္စကိုတော့ နောက်ပိုင်းမှာ ln command ကို သင်တဲ့အခါ ရှင်းပါမယ်။ အခုလောလောဆယ်တော့ ls command ရဲ့ -F option နဲ့ပတ်သတ်တာကိုပဲ အာရုံစိုက်ကြရအောင်။

-F option က * နဲ့ @ အမှတ်အသားတွေ အပြင်၊ အောက်ပါ အမှတ်အသားတွေကိုလဲ ဖိုင်အမျိုးအစားပေါ်မူတည်ပြီး ပြပေးပါလိမ့်မယ်။

/ ဖိုလ်ဒါ
| named pipe ဖိုင်အမျိုးအစား။ FIFO ဖိုင်လို့လဲ ခေါ်တယ်။ နောက်ပိုင်းမှာ အသေးစိတ်ရှင်းပြမယ်။
= socket ဖိုင်
> door ဖိုင်အမျိုးအစားပါ။ Linux မှာ မသုံးပါဘူး။ Sun/Solaris Unix system မှာသုံးတယ်။

ဖွက်ထားတဲ့ ဖိုင်တွေကို ကြည့်ချင်ရင် ls command ရဲ့ option တခုဖြစ်တဲ့ -a သုံးတယ်။

In [5]:

```
ls -a
```

```
.                echo          .ipynb_checkpoints        otest
..               fileA         item-list                 otest.tag
all-vi-info.txt  fileB         linux-commands.ipynb      otest.word
data1            .fileC        linux-commands.ipynb.bak  screen
data2            fmt.out       ls.out                    sorted-names
data3            folderA       my-news                   wildcard
data4            folder-athit  newfile
data5            folder-new    news
```

အထက်မှာ မြင်ခဲ့ရတာနဲ့ မတူတာက "." ရယ်၊ ".." ရယ်၊ ".fileC" ရယ် ဆိုတာကို သတိပြုမိလိမ့်မယ်။
. က လက်ရှိရောက်နေတဲ့ path ကို ညွှန်းတဲ့ အခါမှာ သုံးတယ်။
.. က လက်ရှိရောက်နေတဲ့ path ကနေ အထက်တဆင့်မှာရှိတဲ့ path ကိုညွှန်းတဲ့ အခါမှာ အသုံးပြုတယ်။ (နောက်ပိုင်းမှာ cd command နဲ့ အတူရှင်းပြမယ်)
.fileC နဲ့ .ipynb_checkpoints ဖိုင်က ဖွက်ထားတဲ့ ဖိုင်ပါ။ အများသောအားဖြင့် ဖွက်ထားတဲ့ ဖိုင်တွေက application၊ shell တို့ရဲ့ setting ဖိုင်တွေ ဖြစ်ကြပါတယ်။
Linux OS မှာက "." နဲ့စတဲ့ ဖိုင်နာမည်တွေဆိုရင် ls command က ပြမပေးပါဘူး။ မှားပြီး ဖျက်မိတာမျိုး၊ ဝင်ပြင်ရေးမိတဲ့ အမှားတွေကို လျော့နည်းစေဖို့ ရည်ရွယ်ပါတယ်။

ဖိုင်တွေ၊ ဖိုလ်ဒါတွေ ရဲ့ information ကိုအသေးစိတ်ပြပေးစေချင်ရင် ls command ကို -l option နဲ့တွဲရိုက်ကြည့်ပါ။

In [6]:

```
ls -l
```

```
total 1272
-rw-rw-r-- 1 lar lar  16937  8月 11 11:14 all-vi-info.txt
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data1
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data2
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data3
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data4
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data5
-rw-rw-r-- 1 lar lar      0  8月 11 11:14 echo
-rw-rw-r-- 1 lar lar     67  8月 11 17:48 fileA
-rw-rw-r-- 1 lar lar     61  8月 11 11:14 fileB
-rw-rw-r-- 1 lar lar   3117  8月 11 11:14 fmt.out
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 folderA
drwxrwxr-x 2 lar lar   4096  8月 11 18:04 folder-athit
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 folder-new
-rw-rw-r-- 1 lar lar     51  8月 11 17:46 item-list
-rw-rw-r-- 1 lar lar 237013  8月 11 18:02 linux-commands.ipynb
-rw-rw-r-- 1 lar lar 202186  8月 11 11:14 linux-commands.ipynb.bak
-rw-rw-r-- 1 lar lar    177  8月 11 11:14 ls.out
-rw-rw-r-- 1 lar lar   3117  8月 11 11:14 my-news
-rw-rw-r-- 1 lar lar      0  2月  3  2015 newfile
-rw-rw-r-- 1 lar lar   4246  8月 11 11:14 news
-rw-r--r-- 1 lar lar 377269  8月 11 11:14 otest
-rw-r--r-- 1 lar lar  73420  8月 11 11:14 otest.tag
-rw-r--r-- 1 lar lar 303849  8月 11 11:14 otest.word
drwxrwxr-x 2 lar lar   4096  8月 11 18:01 screen
-rw-rw-r-- 1 lar lar     68  8月 11 11:14 sorted-names
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 wildcard
```

ဘာတွေပြနေသလဲ ဆိုတော့

-rw-rw-r-- 1 lar lar 67 8月 11 11:14 fileA ဆိုတဲ့ စာကြောင်းမှာ
ပထမဆုံး စာလုံး "-" (dash) က ပုံမှန်ဖိုင် လို့ဆိုလိုတယ်။
အဲဒီနေရာမှာ "d" ဆိုရင် directory (or) folder လို့ဆိုလိုတယ်။
rw-rw-r-- (စာလုံး ၉လုံး) က ဖိုင်နဲ့ ပတ်သက်တဲ့ permission တွေကို ပြပေးတာ။
ဘယ်သူက၊ ဘယ်အုပ်စုက ဒီဖိုင်ကို ဖတ်လို့ရတယ် ဆိုတာကို "r" နဲ့၊ ဝင်ပြင်လို့ရတယ် ဆိုတာကို "w" နဲ့၊ ပရိုဂရမ်အနေနဲ့ execute (or) run လုပ်လို့ရတယ် ဆိုတာကို "x" သင်္ကေတနဲ့ ပြပေးလိမ့်မယ်။
1 ဆိုတာက link ဘယ်နှစ်ခုရှိနေသလဲ ဆိုတာကို ပြပေးတာ။
ပထမဆုံးတွေ့ရတဲ့ lar က ဖိုင်ရဲ့ပိုင်ရှင်က ဘယ်သူလဲ (သို့) ဘယ် acount name လဲ ဆိုတာကို ပြပေးတာ။
ဒုတိယတွေ့ရတဲ့ lar က ဘယ်အုပ်စု (group) နဲ့ဆိုင်တယ် ဆိုတာကိုပြပေးတာ။
67 ဆိုတာက ဖိုင်ရဲ့အရွယ်အစား (size) ကို bit နဲ့ပြပေးတာ။
ဥပမာ 1 KB = 1024 bits, 1 MB = 1024 KB။

8月 11 11:14 ဆိုတာက ဖိုင်ကို နောက်ဆုံးပြင်ခဲ့တဲ့ လ၊ ရက်၊ နာရီကို ပြေနေတာ။
မပြောလဲ မြင်မှာပါ။ fileA ဆိုတာက ဖိုင်နာမည်ပါ။

size ကြီးတဲ့ အစီအစဉ်အတိုင်း ပြပေးစေချင်ရင် option -S ကိုသုံးတယ်။
ဖိုင် Size အကြီးဆုံးက ထိပ်ဆုံးမှာ ရှိပါလိမ့်မယ်။

In [7]:

```
ls -lS
```

```
total 1272
-rw-r--r-- 1 lar lar 377269  8月 11 11:14 otest
-rw-r--r-- 1 lar lar 303849  8月 11 11:14 otest.word
-rw-rw-r-- 1 lar lar 237013  8月 11 18:02 linux-commands.ipynb
-rw-rw-r-- 1 lar lar 202186  8月 11 11:14 linux-commands.ipynb.bak
-rw-r--r-- 1 lar lar  73420  8月 11 11:14 otest.tag
-rw-rw-r-- 1 lar lar  16937  8月 11 11:14 all-vi-info.txt
-rw-rw-r-- 1 lar lar   4246  8月 11 11:14 news
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data1
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data2
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data3
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data4
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data5
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 folderA
drwxrwxr-x 2 lar lar   4096  8月 11 18:04 folder-athit
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 folder-new
drwxrwxr-x 2 lar lar   4096  8月 11 18:01 screen
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 wildcard
-rw-rw-r-- 1 lar lar   3117  8月 11 11:14 fmt.out
-rw-rw-r-- 1 lar lar   3117  8月 11 11:14 my-news
-rw-rw-r-- 1 lar lar    177  8月 11 11:14 ls.out
-rw-rw-r-- 1 lar lar     68  8月 11 11:14 sorted-names
-rw-rw-r-- 1 lar lar     67  8月 11 17:48 fileA
-rw-rw-r-- 1 lar lar     61  8月 11 11:14 fileB
-rw-rw-r-- 1 lar lar     51  8月 11 17:46 item-list
-rw-rw-r-- 1 lar lar      0  8月 11 11:14 echo
-rw-rw-r-- 1 lar lar      0  2月  3  2015 newfile
```

ဖိုင်ရဲ့size ငယ်စဉ်ကြီးလိုက် အစီအစဉ်နဲ့ စီပြပေးစေချင်ရင် -r (reverse) ကို သုံးပါတယ်။ option အပြည့်အစုံကတော့ -lSr ပါ။

In [8]:

```
ls -lSr
```

```
total 1272
-rw-rw-r-- 1 lar lar      0  2月  3  2015 newfile
-rw-rw-r-- 1 lar lar      0  8月 11 11:14 echo
-rw-rw-r-- 1 lar lar     51  8月 11 17:46 item-list
-rw-rw-r-- 1 lar lar     61  8月 11 11:14 fileB
-rw-rw-r-- 1 lar lar     67  8月 11 17:48 fileA
-rw-rw-r-- 1 lar lar     68  8月 11 11:14 sorted-names
-rw-rw-r-- 1 lar lar    177  8月 11 11:14 ls.out
-rw-rw-r-- 1 lar lar   3117  8月 11 11:14 my-news
-rw-rw-r-- 1 lar lar   3117  8月 11 11:14 fmt.out
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 wildcard
drwxrwxr-x 2 lar lar   4096  8月 11 18:01 screen
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 folder-new
drwxrwxr-x 2 lar lar   4096  8月 11 18:04 folder-athit
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 folderA
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data5
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data4
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data3
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data2
drwxrwxr-x 2 lar lar   4096  8月 11 11:14 data1
-rw-rw-r-- 1 lar lar   4246  8月 11 11:14 news
-rw-rw-r-- 1 lar lar  16937  8月 11 11:14 all-vi-info.txt
-rw-r--r-- 1 lar lar  73420  8月 11 11:14 otest.tag
-rw-rw-r-- 1 lar lar 202186  8月 11 11:14 linux-commands.ipynb.bak
-rw-rw-r-- 1 lar lar 237013  8月 11 18:02 linux-commands.ipynb
-rw-r--r-- 1 lar lar 303849  8月 11 11:14 otest.word
-rw-r--r-- 1 lar lar 377269  8月 11 11:14 otest
```

ဖိုင်ရဲ့အရွယ်အစား (size) တွေကို လူကဖတ်လို့လွယ်တဲ့ format (ဥပမာ၊ KB, MB, GB, TB) နဲ့ ပြပေးစေချင်ရင် -h option ကိုသုံးပါ။

In [9]:

```
ls -lh
```

```
total 1.3M
-rw-rw-r-- 1 lar lar  17K  8月 11 11:14 all-vi-info.txt
drwxrwxr-x 2 lar lar 4.0K  8月 11 11:14 data1
drwxrwxr-x 2 lar lar 4.0K  8月 11 11:14 data2
drwxrwxr-x 2 lar lar 4.0K  8月 11 11:14 data3
drwxrwxr-x 2 lar lar 4.0K  8月 11 11:14 data4
drwxrwxr-x 2 lar lar 4.0K  8月 11 11:14 data5
-rw-rw-r-- 1 lar lar    0  8月 11 11:14 echo
-rw-rw-r-- 1 lar lar   67  8月 11 17:48 fileA
-rw-rw-r-- 1 lar lar   61  8月 11 11:14 fileB
-rw-rw-r-- 1 lar lar 3.1K  8月 11 11:14 fmt.out
drwxrwxr-x 2 lar lar 4.0K  8月 11 11:14 folderA
drwxrwxr-x 2 lar lar 4.0K  8月 11 18:04 folder-athit
drwxrwxr-x 2 lar lar 4.0K  8月 11 11:14 folder-new
-rw-rw-r-- 1 lar lar   51  8月 11 17:46 item-list
-rw-rw-r-- 1 lar lar 232K  8月 11 18:02 linux-commands.ipynb
-rw-rw-r-- 1 lar lar 198K  8月 11 11:14 linux-commands.ipynb.bak
-rw-rw-r-- 1 lar lar  177  8月 11 11:14 ls.out
-rw-rw-r-- 1 lar lar 3.1K  8月 11 11:14 my-news
-rw-rw-r-- 1 lar lar    0  2月  3  2015 newfile
-rw-rw-r-- 1 lar lar 4.2K  8月 11 11:14 news
-rw-r--r-- 1 lar lar 369K  8月 11 11:14 otest
-rw-r--r-- 1 lar lar  72K  8月 11 11:14 otest.tag
-rw-r--r-- 1 lar lar 297K  8月 11 11:14 otest.word
drwxrwxr-x 2 lar lar 4.0K  8月 11 18:01 screen
-rw-rw-r-- 1 lar lar   68  8月 11 11:14 sorted-names
drwxrwxr-x 2 lar lar 4.0K  8月 11 11:14 wildcard
```

-lt option နဲ့ ဖိုင်တွေ၊ ဖိုလ်ဒါတွေကို အချိန်နဲစီခိုင်းလို့ရပါတယ်။
t option က နောက်ဆုံးပြင်ခဲ့၊ ရေးခဲ့တဲ့ဖိုင်ကို ထိပ်ဆုံးမှာထားပေးပြီး စီပေးပါလိမ့်မယ်။
အောက်ပါ command က ကိုယ်ဒေါင်းလုဒ် လုပ်ခဲ့တဲ့ ဖိုင်တွေကို အချိန်နဲ့ စီကြည့်တဲ့ ဥပမာပါ။
ပုံမှန်အားဖြင့် (default setting) က အင်တာနက်ကနေ ဒေါင်းလုဒ်လုပ်လိုက်တဲ့ ဖိုင်တွေက ဟုမ်းဖိုလ်ဒါအောက်က Downloads ဆိုတဲ့ ဖိုလ်ဒါအောက်မှာ သိမ်းပါတယ်။
(ဒီနေရာမှာ "~" က လက်ရှိ login ဝင်ထားတဲ့ account ရဲ့ home directory ကိုဆိုလိုပါတယ်။)

In [10]:

```
ls ~/Downloads -lt
```

```
total 922700
-rw-rw-r--  1 lar lar      7370  8月 10 04:20 bash_kernel-master (1).z
ip
drwxrwxr-x  7 lar lar      4096  8月 10 04:12 jupyter-nodejs
-rwxrwxr-x  1 lar lar       222  8月 10 03:52 js-kernel-install.sh
-rw-rw-r--  1 lar lar     71197  8月 10 03:47 jupyter-scala-master.zip
-rw-rw-r--  1 lar lar      7370  8月 10 03:43 bash_kernel-master.zip
-rw-rw-r--  1 lar lar     85041  8月  8 17:49 kaigaikoutei-icdl2017.pd
f
drwxrwxr-x  6 lar lar      4096  8月  6 14:58 ocr-data
-rw-rw-r--  1 lar lar 141808408  8月  6 14:35 ocr-demo-site7.zip
-rw-rw-r--  1 lar lar   1104821  8月  6 14:35 train-tif-box-original.z
ip
-rw-rw-r--  1 lar lar  23875368  8月  6 14:35 icu4c-52_1-src.tgz
-rw-rw-r--  1 lar lar   6126586  8月  6 14:35 jTessBoxEditor-my-rule1.
zip
drwxrwxr-x 31 lar lar      4096  8月  6 13:07 tesseract-master
-rw-rw-r--  1 lar lar  43779562  8月  6 13:06 opencv_python_examples-m
aster.zip
drwxrwxr-x  8 lar lar      4096  8月  6 12:30 leptonica-1.74.4
-rw-rw-r--  1 lar lar  12349877  8月  6 12:29 leptonica-1.74.4.tar.gz
-rw-rw-r--  1 lar lar   4524383  8月  6 10:56 tesseract-master.zip
drwxrwxr-x  4 lar lar      4096  8月  6 10:52 tesseract-ocr.github.io-
master
-rw-rw-r--  1 lar lar  43793679  8月  6 10:52 tesseract-ocr.github.io-
master.zip
drwxrwxr-x  3 lar lar      4096  8月  4 18:47 bash_kernel-master
drwxrwxr-x  2 lar lar      4096  8月  3 15:29 ebook
-rw-rw-r--  1 lar lar  40686466  8月  1 22:34 introduction_to_ml_with_
python-master.zip
-rw-rw-r--  1 lar lar 523283080  8月  1 21:25 Anaconda3-4.4.0-Linux-x8
6_64.sh
-rw-rw-r--  1 lar lar  51327604  7月 16 15:36 skypeforlinux-64.deb
drwxrwxr-x 11 lar lar      4096  7月  5 04:11 jupyter-scala-master
-rw-rw-r--  1 lar lar     67600  5月 21 21:42 ctest2.nopipe.word
-rw-rw-r--  1 lar lar   3737012  4月 29 09:33 1505.00687.pdf
-rw-rw-r--  1 lar lar    479931  4月 19 16:29 flexcrfs.pdf
-rw-------  1 lar lar    194870  4月 19 16:25 Introduction to Conditio
nal Random Fields.pdf
-rw-rw-r--  1 lar lar  47321450  4月 19 14:14 google-chrome-stable_cur
rent_amd64.deb
-rw-rw-r--  1 lar lar     47389  4月 12 21:53 myanmar3.zip
drwx------  3 lar lar      4096 10月 31  2015 jTessBoxEditor-1.4-src-e
dit-myanmar-rule1-only
```

လက်ရှိ နေရာအောက်မှာရှိတဲ့ "ဖိုလ်ဒါတွေကိုပဲ" ပြပေးပါဆိုရင် -d option နဲ့ */ pattern ကို တွဲသုံးပါတယ်။

In [11]:

```
ls -d */
```

```
data1/  data3/  data5/    folder-athit/  screen/
data2/  data4/  folderA/  folder-new/    wildcard/
```

-R option နဲ့ ဖိုလ်ဒါရဲ့အောက်မှာရှိနေတဲ့ ဖိုင်တွေ၊ ဖိုလ်ဒါ အဆင့်ဆင့်တွေ အားလုံးကို ပြခိုင်းလို့ ရပါတယ်။

ဥပမာ ~/Pictures/ ရဲ့အောက်မှာ ရှိသမျှ ဖိုင်တွေ၊ ဖိုလ်ဒါတွေကို ပြခိုင်းစေချင်ရင် အောက်ပါအတိုင်း ရိုက်ကြည့်ပါ။
(ဒီနေရာမှာ "~" က လက်ရှိ login ဝင်ထားတဲ့ account ရဲ့ home directory ကိုဆိုလိုပါတယ်။)
လက်ရှိ ကျွန်တော် သုံးနေတဲ့ ကွန်ပြူတာ ရဲ့ဟုမ်းဖိုလ်ဒါအောက်က Pictures ဖိုလ်ဒါအောက်မှာ layout ဆိုတဲ့ ဖိုလ်ဒါကရှိနေတော့ အဲဒီအောက်မှာ ရှိတဲ့ ဖိုင်တွေကို ပြသပေးပါတယ်။
လောလောဆယ် လက်ရှိမှာရှိတဲ့ ဖိုင်နှစ်ဖိုင်ဖြစ်တဲ့ kkg-ver0.1.pdf ဖိုင်နဲ့ Selection_001.png ဖိုင်ကို ပြပေးပါလိမ့်မယ်။
တကယ်လို့ အဲဒီ layout ဆိုတဲ့ ဖိုလ်ဒါအောက်မှာ နောက်တစ်ဆင့် ဖိုလ်ဒါတွေသာရှိခဲ့ရင်၊ အဲဒီဖိုလ်ဒါအောက်မှာရှိတဲ့ ဖိုင်တွေ၊ ဖိုလ်ဒါတွေကို ဆင့်ကဲဆင့်ကဲ ပြသပေးမှာဖြစ်ပါတယ်။

In [12]:

```
ls -R ~/Pictures/
```

```
/home/lar/Pictures/:
Choose an input source_003.png  Menu_005.png          selecting-kkg.pn
g
ChooseAnInputSource.jpg         selecting-kkg-eg.jpg  Selection_006.pn
g
layout                          selecting-kkg-eg.png  Text Entry_002.p
ng
Menu_004.png                    selecting-kkg.jpg     Text-Entry-Dbox.
jpg

/home/lar/Pictures/layout:
kkg-ver0.1.pdf  Selection_001.png
```

2. pwd (print name of current/working directory)

လက်ရှိ ရောက်နေတဲ့ ဖိုလ်ဒါကို သိချင်ရင်

In [13]:

```
pwd
```

```
/home/lar/linux-cmd
```

Linux မှာ ဖိုလ်ဒါတခုနဲ့ တခုအကြားကို / (forward slash) နဲ့ ခွဲပြီးပြတယ်။

တစ်ခုသိထားစေချင်တာက linux command တွေမှာက အင်္ဂလိပ်စာလုံး အကြီး၊ အသေးကို ခွဲခြားပြီးသုံးတယ်။ အဲဒါကြောင့် pwd ကို ရိုက်တဲ့ အခါမှာ စာလုံးအားလုံးက အသေးဖြစ်ရမယ်။ တကယ်လို့ အခုအချိန်ထိသင်ပေးခဲ့တဲ့ command နှစ်ခုဖြစ်တဲ့ "ls" နဲ့ "pwd" ကို LS, lS, PWD, pWd စသည်ဖြင့်ရိုက်ရင် အလုပ်မလုပ်ပါဘူး။ error message ပေးပါလိမ့်မယ်။

3. cd (change the shell working directory)

ဖိုလ်ဒါ တခုကနေ နောက်ဖိုလ်ဒါ တခုကို ရွှေ့မယ်ဆိုရင် သုံးတဲ့ command ပါ။
ဥပမာ လက်ရှိရောက်နေတဲ့ နေရာအောက်မှာ ဖိုလ်ဒါနာမည် folderA ရှိတယ်။ အဲဒီ ဖိုလ်ဒါဆီကို ရွှေ့ပေးပါဆိုရင်။

In [14]:

```
cd folderA
```

လက်ရှိရောက်နေတဲ့ folder path ကနေ ကိုယ့်ရဲ့ home ဖိုလ်ဒါဆီ ကို ရွှေ့ချင်ရင် ~ (tilde) option ကို သုံးပါတယ်။

In [15]:

```
cd ~
```

ပြီးတော့ လက်ရှိရောက်နေတဲ့ ဖိုလ်ဒါနေရာကို confirm လုပ်ဖို့အတွက်၊ အထက်မှာ ပြောခဲ့တဲ့ pwd command သုံးကြည့်မယ်။

In [16]:

```
pwd
```

```
/home/lar
```

လက်ရှိရောက်နေတဲ့ /home/hb/ နေရာကနေ့ အထက်တဆင့် path ကိုတက်ချင်တဲ့ အခါမှာ .. ကို သုံးပြီးတက်နိုင်။

In [17]:

```
cd ..
```

In [18]:

```
pwd
```

```
/home
```

- option ကို သုံးပြီး နောက်ဆုံးဝင်ခဲ့တဲ့ ဖိုလ်ဒါနေရာ (path) ဆီကို ပြန်ရွေ့လို့ရပါတယ်။

အောက်ပါဥပမာက ပထမ လက်ရှိရှိနေရာတဲ့ နေရာကနေ ရှည်လျားတဲ့ ဖိုလ်ဒါ တခုအောက် (/usr/share/X11/xkb/geometry/) ကို ဝင်ပါတယ်။ အဲဒီမှာ ရှိတဲ့ ဖိုင်တွေကို ls နဲ့ list လုပ်ကြည့်ပါတယ်။

In [19]:

```
cd /usr/share/X11/xkb/geometry/
```

In [20]:

```
ls
```

```
amiga    digital_vndr  hp         microsoft  pc        sony      typem
atrix
ataritt  everex        keytronic  nec        README    sun       winbo
ok
chicony  fujitsu       kinesis    nokia      sanwa     teck
dell     hhk           macintosh  northgate  sgi_vndr  thinkpad
```

ပြီးတော့ ကိုယ်ရဲ့ home folder အောက်ကို cd ~ နဲ့ သွားပါတယ်။

In [21]:

```
cd ~
```

မြင်သာအောင် pwd နဲ့ home folder အောက်ကိုရောက်နေတာကို ရိုက်ပြထားတာပါ။

In [22]:

```
pwd
```

```
/home/lar
```

စိတ်ကူးပြောင်းသွားပြီး ခုနက နောက်ဆုံးရှိနေခဲ့တဲ့ဖိုလ်ဒါအောက်ဖြစ်တဲ့ /usr/share/X11/xkb/geometry/ ဆီကို ပြန်ဝင်ချင်တဲ့ အခါမှာ ဖိုလ်ဒါနာမည်တွေကို ပြန်ရိုက်စရာမလိုပဲ - (dash option) ကိုသုံးလို့ရကြောင်းကို လုပ်ပြတာပါ။ တကယ်တမ်း command တွေကို သုံးပြီး အလုပ်လုပ်တဲ့ အခါမှာ မြန်ဆန်ဖို့အတွက် ဒီ - option က အရမ်းကို အသုံးဝင်ပါတယ်။ အထူးသဖြင့် ရှည်လျားတဲ့ ဖိုလ်ဒါ path တွေ တစ်ခုက နေ တစ်ခုကို ရွှေ့ပြောင်းပြီး အလုပ်လုပ်တဲ့ အခါမျိုးမှာပါ။ မှတ်ထားသင့်ပါတယ်။

In [23]:

```
cd -
```

```
/usr/share/X11/xkb/geometry
```

4. cat (concatenate files and print on the standard output)

cat command က ဖိုင်တဖိုင်ရဲ့အထဲမှာ ရှိတဲ့ စာကြောင်းတွေ အားလုံးကို ရိုက်ထုတ်ပြဖို့ သုံးပါတယ်။
cat command ကို မသုံးခင်၊ အရင်ဆုံး linux command တွေကို သင်ကြားဖို့အတွက် ကျွန်တော်က ပြင်ဆင်ထားတဲ့ path ဆီကို ရွှေ့မယ်။

In [24]:

```
cd ~/linux-cmd
```

In [25]:

```
pwd
```

```
/home/lar/linux-cmd
```

ဘာဖိုင်တွေ၊ ဘာဖိုလ်ဒါတွေ ရှိသလဲ ဆိုတာကို ls နဲ့အရင်ကြည့်မယ်။
ဒီတခါတော့ option ကို -1X ကိုသုံးကြည့်မယ်။
-1X နဲ့ဆိုရင် ရှိတဲ့ ဖိုင်နာမည်၊ ဖိုလ်ဒါနာမည်တွေကို တစ်ကြောင်းချင်းစီ ရိုက်ပြလိမ့်မယ်။

In [26]:

```
ls -1X
```

data1
data2
data3
data4
data5
echo
fileA
fileB
folderA
folder-athit
folder-new
item-list
my-news
newfile
news
otest
screen
sorted-names
wildcard
linux-commands.ipynb.bak
linux-commands.ipynb
fmt.out
ls.out
otest.tag
all-vi-info.txt
otest.word

ဒီနေရာမှာ -1 က စာကြောင်းတစ်ကြောင်းမှာ ဖိုင်တစ်ဖိုင်စီ ရိုက်ပြတဲ့အလုပ် (list one file per line) ကို လုပ်ပေးတဲ့ ls command ရဲ့ option ပါ။
-X (Capital X option) ကတော့ ဖိုင်တွေရဲ့ extension နဲ့ အက္ခရာစဉ်အလိုက်စီပေးပါတယ်။

Extension ဆိုတာက ဖိုင်နာမည်တွေရဲ့ နောက်ဆုံးမှာ ရှိနေတဲ့ dot နဲ့ခွဲပြီးရေးထားတဲ့ အပိုင်းတွေကို ဆိုလိုပါတယ်။
ဥပမာ PDF ဖိုင်ဆိုရင် .pdf၊ ပုံဖိုင်အမျိုးအစားတစ်မျိုးဖြစ်တဲ့ .jpeg၊ bash shell script ဖိုင်ဖြစ်တဲ့ .sh စတာမျိုးတွေပါ။

cat command ကိုသုံးပြီး fileA ရဲ့အထဲမှာ ဘာတွေရေးထားလဲ ဆိုတာကို ရိုက်ထုပ်ကြည့်မယ်။

In [27]:

```
cat fileA
```

```
Mingalar bar!
I am fileA.
I was born in April.
My blood type is O.
```

ဖိုင်ထဲမှာရှိတဲ့ စာကြောင်းတွေကို ရိုက်ပြတဲ့ အခါမှာ လိုင်းနံပါတ်တွေ တပ်ပြီးရိုက်ပြစေချင်ရင် -n option ကိုသုံးပါ။

In [28]:

```
cat -n fileA
```

```
     1  Mingalar bar!
     2  I am fileA.
     3  I was born in April.
     4  My blood type is O.
```

cat command ရဲ့ အသုံးဝင်တဲ့ option နှစ်ခုဖြစ်တဲ့ -T နဲ့ -E ကို သုံးပြဖို့အတွက် item-list ဆိုတဲ့ ဖိုင်အသစ်တစ်ဖိုင်ကို echo command ကို သုံးပြီး အောက်ပါအတိုင်းဆောက်ပါမယ်။ ဒီနေရာမှာ "\t" က tab ကီး "\n" က စာကြောင်းတစ်ကြောင်း အောက်ကိုဆင်းဖို့အတွက် သုံးတဲ့ escape စာလုံးတွေပါ။

In [29]:

```
echo -e "No\tItem\tStock\n1\tpencil\t10\n2\truler\t1000\n3\teraser\t42" > item-list
```

အခုဆောက်ခဲ့တဲ့ item-list ထဲမှာ ဘယ်လိုစာကြောင်းတွေရှိနေသလဲဆိုတာကို cat command နဲ့ ရိုက်ကြည့်ပါမယ်။

In [30]:

```
cat ./item-list
```

```
No      Item    Stock
1       pencil  10
2       ruler   1000
3       eraser  42
```

အထက်ပါအတိုင်း No ရယ် Item ရယ် Stock ရယ်ဆိုပြီး ကော်လံသုံးခု ခွဲထားပြီး၊ ပစ္စည်း သုံးမျိုးဖြစ်တဲ့ pencil, ruler, eraser ရဲ့ ရှိနေတဲ့ အရေအတွက်ကို သိမ်းထားတာတွေ့ရပါလိမ့်မယ်။ လက်တွေ့ item-list လို text file တွေကို၊ ကိုယ်လုပ်မဲ့ အလုပ်ပေါ် မူတည်ပြီးတော့ ပုံမှန်အားဖြင့် မမြင်နိုင်တဲ့ tab ကီးတွေ၊ စာကြောင်း အဆုံးသင်္ကေတ တွေကို စစ်ဆေးကြည့်ဖို့၊ မြင်ရအောင် ရိုက်ထုတ်ချင်တဲ့အခါမှာ cat command နဲ့ တကွ -T, -E option တွေကို အသုံးပြုပါတယ်။

-T က tab ကီးကိုမြင်ရဖို့၊ -E က စာကြောင်းအဆုံးသင်္ကေတကို မြင်ရဖို့သုံးပါတယ်။

In [31]:

```
cat -TE ./item-list
```

```
No^IItem^IStock$
1^Ipencil^I10$
2^Iruler^I1000$
3^Ieraser^I42$
```

အထက်ပါ output မှာမြင်ရတဲ့အတိုင်း၊ ^I ဆိုတာက tab ကီးကို ကိုယ်စားပြုပါတယ်။ $ စာလုံးက စာကြောင်းအဆုံးကို ကိုယ်စားပြုပါတယ်။ ဒီသင်္ကေတတွေကို မျက်လုံးနဲ့ ဖိုင်တွေကိုစစ်ကြည့်ဖို့ တင်မကပဲ၊ ပရိုဂရမ်နဲ့ ဖိုင်ကိုဖတ်ပြီး စစ်ဆေးကြည့်တဲ့ အခါမှာလည်း အသုံးပြုပါတယ်။

5. file (determine file type)

ဖိုင်တဖိုင်က ဘာဖိုင်လဲ၊ ဘယ်လို ဖိုင်အမျိုးအစားလဲ ဆိုတာကို သိချင်ရင် file ဆိုတဲ့ command ကိုသုံးတယ်။
ဖိုင်ရဲ့ အမျိုးအစားဆိုတာက စာသားတွေအနေနဲ့ သိမ်းထားတဲ့ text file လား၊ အလုပ်လုပ်ခိုင်းလို့ရတဲ့ (အင်္ဂလိပ်လိုပြောရင် run/executable လုပ်လို့ရတဲ့) ပရိုဂရမ်ဖိုင်လား၊ ပရိုဂရမ်ဖိုင်ဆိုရင် အဲဒီပရိုဂရမ်ရဲ့ ဗားရှင်းနဲ့ ပတ်သက်တဲ့ အချက်အလက် စတာတွေကို file command ကို သုံးပြီး ရယူနိုင်ပါတယ်။

In [32]:

```
file fileA
```

```
fileA: ASCII text
```

fileA က ASCII (American Standard Code for Information Interchange) text ဖိုင် အမျိုးအစားဖြစ်ကြောင်းကို သိရပါတယ်။
လက်ရှိကွန်ပြူတာမှာ ရှိတဲ့ ဖိုင်တချို့ကို သုံးပြီး file command ရဲ့ output တချို့ကို ဥပမာ အနေနဲ့ ပြပါမယ်။

In [33]:

```
file /bin/cat
```

```
/bin/cat: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamic
ally linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 2.
6.32, BuildID[sha1]=2267d831560007f67fa4388d830192fd89861061, stripped
```

In [34]:

```
file /etc/hostname
```

```
/etc/hostname: ASCII text
```

ဖိုလ်ဒါရဲ့ path ကိုပေးရင် file command က ဖိုလ်ဒါဖြစ်ကြောင်းကို directory ဆိုပြီး အောက်ပါအတိုင်း ဖော်ပြပေးပါလိမ့်မယ်။

In [35]:

```
file /home/
```

```
/home/: directory
```

In [36]:

```
file otest
```

```
otest: UTF-8 Unicode text, with very long lines
```

6. head (output the first part of files)

ဖိုင်တဖိုင်ထဲမှာရှိတဲ့ စာကြောင်းတွေကို ထိပ်ဆုံးအပိုင်းကနေ ရိုက်ထုတ်ပြပေးဖို့ သုံးတဲ့ command ဖြစ်ပါတယ်။
ဥပမာ otest ဖိုင်ရဲ့ထိပ်ဆုံး စာကြောင်း ၁၀ကြောင်းကို ရိုက်ပြခိုင်းချင်ရင်

In [37]:

```
head otest
```

```
အနောက်တိုင်း/n ဒဿနိကဗေဒ/n မူ/conj သေးလီး/n ဆို/v သော/part ဂရိ/n ပညာရှိ/n ကြီး/pa
rt မှ/ppm စတင်/v ပေါက်ပွား/v လာ/part လေ/part သည်/ppm ။/punc
အများအားဖြင့်/adv သို့မဟုတ်/conj အားလုံး/pron မ/part_neg ဟုတ်/v သော/part စစ်/n အ
စိုးရ/n ၏/ppm အဖွဲ့ဝင်/n အများစု/pron သည်/ppm ဖျက်သိမ်း/v လိုက်/part သည့်/part နဝတ/
abb အဖွဲ့/n ထဲ/ppm မှ/ppm အဖွဲ့ဝင်/n များ/part ပင်/part ဖြစ်/v သည်/ppm ။/punc
အမျိုးသား/n ဒီမိုကရေစီ/n အဖွဲ့ချုပ်/n မှ/ppm အနိုင်/n ရရှိ/v ကြောင်း/part နိုင်ငံတော်/n ငြိမ်ပိ/ab
b က/ppm ကြေညာ/v သော်လည်း/conj အာဏာ/n လွှဲပြောင်းရေး/n နှင့်/ppm ပတ်သက်/v ၍/con
j တိတိကျကျ/adv မ/part_neg ကြေညာ/v သေး/part ပေ/part ။/punc
အယ်လဖာ/n ရောင်ခြည်/n ကို/ppm တစ်ခါတစ်ရံ/adv အယ်လဖာ/n မြူန်/n ဟူ၍/part လည်း/par
t ခေါ်/v တတ်/part ကြ/part သည်/ppm ။/punc
အဝတ်အစား/n နဲ့/ppm ပတ်သက်/v ပြီး/conj အမြင်ရှိ/v တယ်/ppm နော်/part ။/punc
အိတ်/n တွေ/part ခဏ/n ထား/v လို့/part ရ/v တဲ့/part ရုံးခန်း/n က/ppm ဘယ်/adj နေ
ရာ/n မှာ/ppm လဲ/part ။/punc
အောက်စဖို့ဒ်/n အင်္ဂလိပ်/n သတ်ပုံ/n ရေးထုံး/n တွင်/ppm '/punc အာရ်/sb '/punc သည်/ppm
 ဗျည်း/n သံရည်/n ကို/ppm ညွှန်း/v ခြင်း/part ဖြစ်/v သည်/ppm ။/punc
အဲဒါ/pron ကို/ppm လမ်းပြ/n မြေပုံ/n ကို/ppm ပြ/v ပြီး/conj သင်ပေး/v ပါ/part ။/punc
အဲဒီ/pron ခလုတ်/n ကို/ppm နှိပ်/v လိုက်/part ရုံ/part ဘဲ/part ။/punc
ဣတိတသ္မာ/n ၊/punc ထို့ကြောင့်/conj နိဂ္ဂမနီယော/n ၊/punc နိဂ္ဂမနီယ/n မည်/ppm ၏/ppm
 ။/punc
```

ဒီ otest ဖိုင်က မြန်မာစာ POS Tagging သုတေသန experiment လုပ်နေစဉ်မှာ၊ ကျွန်တော်ရဲ့တပည့်တယောက်ဖြစ်တဲ့ မဝါဝါက ပြင်ဆင်ထားတဲ့ test ဖိုင်တဖိုင်ပါ။

head command က ဘာ option မှမပေးရင် ရိုက်ခိုင်းတဲ့ ဖိုင်ရဲ့ထိပ်ဆုံးကနေ ၁၀ ကြောင်းကို ရိုက်ပြပါတယ်။
တကယ်လို့ သတ်မှတ်ထားတဲ့ စာကြောင်းရေအရေအတွက်ကိုပဲ ဥပမာ ၁ ကြောင်း၊ ၃ ကြောင်း၊ အကြောင်း ၁၀၀ စသည်ဖြင့် ရိုက်ပြပေးစေချင် ရင် -n option ကို သုံးပါ။

In [38]:

```
head -n 3 otest
```

```
အနောက်တိုင်း/n ဒဿနိကဗေဒ/n မူ/conj သေးလီး/n ဆို/v သော/part ဂရိ/n ပညာရှိ/n ကြီး/pa
rt မှ/ppm စတင်/v ပေါက်ပွား/v လာ/part လေ/part သည်/ppm ။/punc
အများအားဖြင့်/adv သို့မဟုတ်/conj အားလုံး/pron မ/part_neg ဟုတ်/v သော/part စစ်/n အ
စိုးရ/n ၏/ppm အဖွဲ့ဝင်/n အများစု/pron သည်/ppm ဖျက်သိမ်း/v လိုက်/part သည့်/part နဝတ/
abb အဖွဲ့/n ထဲ/ppm မှ/ppm အဖွဲ့ဝင်/n များ/part ပင်/part ဖြစ်/v သည်/ppm ။/punc
အမျိုးသား/n ဒီမိုကရေစီ/n အဖွဲ့ချုပ်/n မှ/ppm အနိုင်/n ရရှိ/v ကြောင်း/part နိုင်ငံတော်/n ငြိမ်ပိ/ab
b က/ppm ကြေညာ/v သော်လည်း/conj အာဏာ/n လွှဲပြောင်းရေး/n နှင့်/ppm ပတ်သက်/v ၍/con
j တိတိကျကျ/adv မ/part_neg ကြေညာ/v သေး/part ပေ/part ။/punc
```

-n လို့ မရိုက်ပဲ "-" တခုတည်းလို့ သုံးပြီး၊ "-" အနောက်ကနေ ရိုက်ပြစေချင်တဲ့ စာကြောင်းရေအရေအတွက်နံပါတ်ကို ပေးပြီးလဲ သုံးနိုင်ပါတယ်။

In [39]:

```
head -1 otest
```

```
အနောက်တိုင်း/n ဒဿနိကဗေဒ/n မူ/conj သေးလီး/n ဆို/v သော/part ဂရိ/n ပညာရှိ/n ကြီး/pa
rt မှ/ppm စတင်/v ပေါက်ပွား/v လာ/part လေ/part သည်/ppm ။/punc
```

7. tail (output the last part of files)

tail command က အထက်မှာ ရှင်းပြခဲ့တဲ့ head command နဲ့ ပြောင်းပြန်ပါ။
ဖိုင်တဖိုင်ကို နောက်ဆုံး အပိုင်းကနေ အကြောင်းရေ အရေအတွက်နဲ့ ရိုက်ပြခိုင်းလို ရတဲ့ command ပါ။

အကြောင်းအရေအတွက်ကို ဘာမှ မကန့်သတ်ပဲ run ရင် default အနေနဲ့ ဖိုင်ရဲ့နောက်ဆုံး ၁၀ ကြောင်းကို ရိုက်ပြပါလိမ့်မယ်။

In [40]:

```
tail ~/linux-cmd/otest.tag
```

```
conj n v part ppm num n ppm n num n conj num n ppm v n part ppm n ppm
 n v n part v ppm v v part part conj v v n ppm num n ppm v part part p
pm punc
n ppm n n n ppm tn n part n n n num n part v conj n n conj n n n ppm t
n n part n n num part v ppm punc
num n ppm n v n v part n conj n n ppm part punc n n ppm n n ppm v part
conj punc n ppm part v conj punc v n n part adv v part part n part v p
art ppm punc
num pron v conj n ppm ppm v ppm punc
num n n n ppm n ppm v part part n n ppm n n part ppm n n n part ppm n
 n n v n n n ppm v v part ppm punc
num ppm pron ppm n n n n n n ppm fw ppm v conj n ppm v part conj v par
t part part ppm punc
num n n part ppm n n ppm fw part v adj part part ppm punc
num n part punc
pron ppm ppm v n part adj n n part ppm adv v part ppm punc
punc n punc n punc n punc n part pron part adv v ppm ppm v part part c
onj n part ppm v part ppm punc
```

အကြောင်းအရေအတွက်ကို -n option နဲ့ ကန့်သတ်ပေးလို့ရပါတယ်။
ဥပမာ ဖိုင်ရဲ့နောက်ဆုံးစာကြောင်း တစ်ကြောင်းကို ပဲ ရိုက်ပေးစေချင်ရင် အောက်ပါအတိုင်း -n 1 ဆိုပြီး option ပေးပြီး run ခိုင်းလို့ရပါတယ်။

In [41]:

```
tail -n 1 ~/linux-cmd/otest.tag
```

```
punc n punc n punc n punc n part pron part adv v ppm ppm v part part c
onj n part ppm v part ppm punc
```

head command မှာတုန်းကပြခဲ့သလိုပါပဲ။ -n option မထည့်ပဲ စာကြောင်းအရေအတွက်ကိုပဲ ပြောပြီး အောက်ပါအတိုင်း run လို့လည်း ရပါတယ်။

In [42]:

```
tail -1 ~/linux-cmd/otest.tag
```

```
punc n punc n punc n punc n part pron part adv v ppm ppm v part part c
onj n part ppm v part ppm punc
```

8. cp (copy files and directories)

ဖိုင်တွေ၊ ဖိုလ်ဒါတွေကို ကော်ပီကူးဖို့ အတွက် သုံးတဲ့ command ပါ။
fileA ကို လက်ရှိဖိုလ်ဒါအောက်မှာပဲ fileZ အဖြစ် ကော်ပီကူးမယ်ဆိုရင်

In [43]:

```
cp fileA fileZ
```

In [44]:

```
ls fileA fileZ -la
```

```
-rw-rw-r-- 1 lar lar 67  8月 11 17:48 fileA
-rw-rw-r-- 1 lar lar 67  8月 11 18:04 fileZ
```

ဖိုင်နာမည်တွေကို ပေးရင်းနဲ့ ဘယ် ဖိုလ်ဒါ(path) ရဲ့အောက်က ဖိုင် ကို ဘယ် ဖိုလ်ဒါအောက် ကို ကေပီကူးပေးပါ ဆိုပြီး ဖိုလ်ဒါတွေရဲ့ လမ်းကြောင်းတွေကို အတိအကျ ညွှန်ကြားပြီးလဲ ခိုင်းနိုင်ပါတယ်။ ဥပမာ အနေနဲ့ ~/paper/pacling2017/anm/ ဖိုလ်ဒါအောက်မှရှိတဲ့ anmpacling-updated.pdf ဖိုင်ကို လက်ရှိ ./ (ဖိုလ်ဒါအောက် ကို) ကော်ပီကူးပေးပါ ဆိုပြီး အောက်ပါအတိုင်း command ပေးနိုင်ပါတယ်။

In [45]:

```
cp ~/bk-dlbox/tool4all/127-135.pdf ./folderA/
```

မှတ်ချက်။ ။ Linux OS မှာ ./ ဆိုတာက လက်ရှိရောက်နေတဲ့ ဖိုလ်ဒါpath ကို ညွှန်ပြီး၊ ../ ဆိုတာက လက်ရှိရှိနေတဲ့ ဖိုလ်ဒါpath ရဲ့အထက်တဆင့်ကို ညွှန်းပါတယ်။ လက်ရှိ ဖိုလ်ဒါကိုကူးတဲ့ အခါမှာ ./ ဆိုတာကို ပြောမနေတော့ပဲ ကိုယ်ကော်ပီကူးပြီး သိမ်းချင်တဲ့ ဖိုင်နာမည်ကိုပဲ တန်းပြောလို့ဖြစ်ပါတယ်။

ဖိုလ်ဒါကို ကော်ပီကူးမယ် ဆိုရင် cp command ကို -r option နဲ့ တွဲသုံးရမယ်။

ဥပမာ folderA/ ကို ကော်ပီကူးမယ် ဆိုရင် folderA-Copy/ အဖြစ်နဲ့ ကော်ပီကူးမယ် ဆိုရင်

In [46]:

```
cp -r ./folderA ./folderA-Copy
```

In [47]:

```
ls -p
```

```
all-vi-info.txt  echo      folderA-Copy/            ls.out     otest.
word
data1/           fileA     folder-athit/            my-news
screen/
data2/           fileB     folder-new/              newfile    sorted
-names
data3/           fileZ     item-list                news       wildca
rd/
data4/           fmt.out   linux-commands.ipynb      otest
data5/           folderA/  linux-commands.ipynb.bak  otest.tag
```

ဒီနေရာမှာ ls command ရဲ့ -p option က ဖိုလ်ဒါတွေကို indicator "/" နဲ့ တွဲပြခိုင်းတာပါ။ ဖိုင်နဲ့ ဖိုလ်ဒါကို ကွဲကွဲပြားပြားမြင်ရအောင်လုပ်တဲ့ option ပါ။ အထူးသဖြင့်တော့ ကာလာနဲ့ မပြအောင် setting လုပ်ထားတဲ့ terminal တွေမှာသုံးလေ့ရှိပါတယ်။

9. mv (move or rename file)

ဖိုင်တွေ ဖိုလ်ဒါတွေကို နာမည်ပြောင်းဖို့ အတွက် (သို့) နေရာတခုကနေ တခြားနေရာတခုကို ပြောင်းရွှေ့ဖို့ ဆိုရင် mv command ကိုသုံးပါတယ်။
ဥပမာ fileZ ကို fileY အဖြစ် နာမည်ပြောင်း ချင်ရင်

In [48]:

```
ls file*
```

```
fileA  fileB  fileZ
```

In [49]:

```
mv fileZ fileY
```

In [50]:

```
ls file*
```

```
fileA  fileB  fileY
```

နဂိုက ရှိနေတဲ့ fileZ က fileY အဖြစ်ပြောင်းသွားတာကို ls command နဲ့ confirm လုပ်ကြည့်ရင်တွေ့ရလိမ့်မယ်။
mv command နဲ့ တွဲသုံးတဲ့ -f option ကိုလဲ သိထားသင့်တယ်။
-f option ကိုပါတွဲသုံးရင် နာမည်ပြောင်းတာတို့ ရွှေ့တာတို့လုပ်တဲ့ အခါမှာ တကယ်လို့ အရင်ရှိနေတဲ့ နာမည်တူ ဖိုင်၊ ဖိုလ်ဒါကို overwrite လုပ်ဖို့ လိုအပ်လာတဲ့ အခါမှာ ဘာမှ confirm လုပ်တာ၊ မေးတာ မလုပ်ပဲ ပြောင်းချသွားပေးလိမ့်မယ်။ တနည်းအားဖြင့်ပြောရရင် အရင်ဖိုင်ရှိနေတာ ငါသိပြီးသား၊ လုပ်သာလုပ်ဆိုပြီး by force နဲ့ mv လုပ်ခိုင်းတာ။
ဥပမာ အနေနဲ့၊ ပရိဂရမ်ထဲကနေ mv လုပ်ခိုင်းတာမျိုးမှာ အရင်ရှိပြီးသားဖိုင်တွေနဲ့ ပတ်သက်ပြီး တွေ့လာတိုင်း ဖြေနေစရာ မလိုအောင်လို့ -f option ကို သုံးတယ်။ မသုံးခင်မှာ သေသေချာချာ စဉ်းစားပါ။

10. rm (remove files or directories)

ဖိုင်တွေ၊ ဖိုလ်ဒါတွေကို ဖျက်ဖို့အတွက် သုံးတဲ့ command ပါ။
ဥပမာ fileY ကို ဖျက်မယ်ဆိုရင်

In [51]:

```
rm fileY
```

ဖျက်ပြီး သွားပြီလား ဆိုတာကို ls command နဲ့ confirm လုပ်ကြည့်တဲ့ အခါမှာ၊ ဖျက်ပြီးသွားကြောင်း၊ fileY က မရှိတော့ကြောင်း တွေ့ရပါလိမ့်မယ်။

In [52]:

```
ls file*
```

```
fileA  fileB
```

ဖိုလ်ဒါကို ဖျက်မယ်ဆိုရင် rm command ကို -r option နဲ့ တွဲသုံးတယ်။
အရင်ဆုံး လက်ရှိရောက်နေတဲ့ ဖိုလ်ဒါအောက်မှာ ဘာဖိုင်တွေ၊ ဘာဖိုလ်ဒါတွေ ရှိသလဲ ls command နဲ့ ရိုက်ပြခိုင်းမယ်။

In [53]:

```
ls -p
```

```
all-vi-info.txt  fileA          item-list                 otest
data1/           fileB          linux-commands.ipynb      otest.tag
data2/           fmt.out        linux-commands.ipynb.bak  otest.word
data3/           folderA/       ls.out                    screen/
data4/           folderA-Copy/  my-news                   sorted-names
data5/           folder-athit/  newfile                   wildcard/
echo             folder-new/    news
```

စောစောက ကော်ပီကူးထားခဲ့တဲ့ folderA-Copy ဆိုတာကို -r option မသုံးပဲ ဖျက်ကြည့်ရအောင်။

In [54]:

```
rm folderA-Copy
```

```
rm: cannot remove 'folderA-Copy': Is a directory
```

directory (သို့) ဖိုလ်ဒါ ဖြစ်နေလို့ rm command က ဖျက်လို့ မရကြောင်း error message ပေးပါလိမ့်မယ်။
-r option ကို သုံးပြီး ဖျက်ကြည့်မယ်။

In [55]:

```
rm -r folderA-Copy
```

In [56]:

```
ls -p
```

```
all-vi-info.txt  fileA          linux-commands.ipynb      otest.tag
data1/           fileB          linux-commands.ipynb.bak  otest.word
data2/           fmt.out        ls.out                    screen/
data3/           folderA/       my-news                   sorted-names
data4/           folder-athit/  newfile                   wildcard/
data5/           folder-new/    news
echo             item-list      otest
```

folderA-Copy/ ကပျက်သွား တာကို တွေ့ရမယ်။
rm command က အရမ်း အန္တရာယ်ရှိတဲ့ command တခု ဖြစ်ပါတယ်။
နောက် Linux OS commandline မှာ မှားဖျက်လိုက်မိတဲ့ ဖိုင်ကို ပြန်ရဖို့ဆိုတာမျိုး မမျှော်လင့်ပါနဲ့။
သတိထားပြီး သုံးပါ။

11. alias (create simple names or abbreviations)

အသုံးများသော command တွေကို ကိုယ်လိုချင်တဲ့ option တွေနဲ့ setting လုပ်ထားပြီး command နာမည်အသစ်အနေနဲ့ သိမ်းပေးနိုင်။

ဥပမာ ls -la (-l က long listing format, -a က all ဆိုတဲ့ option ဖွက်ထားတဲ့ ဖိုင်တွေလဲပြ) command ကို ll ဆိုတဲ့ နာမည်နဲ့ alias လုပ်ထား လို့ရတယ်။ အောက်ပါ အတိုင်း alias လုပ်ကြည့်ပြီး ll command ကို run ကြည့်ပါ။

In [57]:

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

In [58]:

```
ll
```

```
total 1284
drwxrwxr-x 13 lar lar   4096  8月 11 18:05 .
drwxr-xr-x 42 lar lar   4096  8月 11 13:29 ..
-rw-rw-r--  1 lar lar  16937  8月 11 11:14 all-vi-info.txt
drwxrwxr-x  2 lar lar   4096  8月 11 11:14 data1
drwxrwxr-x  2 lar lar   4096  8月 11 11:14 data2
drwxrwxr-x  2 lar lar   4096  8月 11 11:14 data3
drwxrwxr-x  2 lar lar   4096  8月 11 11:14 data4
drwxrwxr-x  2 lar lar   4096  8月 11 11:14 data5
-rw-rw-r--  1 lar lar      0  8月 11 11:14 echo
-rw-rw-r--  1 lar lar     67  8月 11 17:48 fileA
-rw-rw-r--  1 lar lar     61  8月 11 11:14 fileB
-rw-rw-r--  1 lar lar      0  8月 11 11:14 .fileC
-rw-rw-r--  1 lar lar   3117  8月 11 11:14 fmt.out
drwxrwxr-x  2 lar lar   4096  8月 11 11:14 folderA
drwxrwxr-x  2 lar lar   4096  8月 11 18:04 folder-athit
drwxrwxr-x  2 lar lar   4096  8月 11 11:14 folder-new
drwxr-xr-x  2 lar lar   4096  8月 11 11:14 .ipynb_checkpoints
-rw-rw-r--  1 lar lar     51  8月 11 18:04 item-list
-rw-rw-r--  1 lar lar 237013  8月 11 18:02 linux-commands.ipynb
-rw-rw-r--  1 lar lar 202186  8月 11 11:14 linux-commands.ipynb.bak
-rw-rw-r--  1 lar lar    177  8月 11 11:14 ls.out
-rw-rw-r--  1 lar lar   3117  8月 11 11:14 my-news
-rw-rw-r--  1 lar lar      0  2月  3  2015 newfile
-rw-rw-r--  1 lar lar   4246  8月 11 11:14 news
-rw-r--r--  1 lar lar 377269  8月 11 11:14 otest
-rw-r--r--  1 lar lar  73420  8月 11 11:14 otest.tag
-rw-r--r--  1 lar lar 303849  8月 11 11:14 otest.word
drwxrwxr-x  2 lar lar   4096  8月 11 18:01 screen
-rw-rw-r--  1 lar lar     68  8月 11 11:14 sorted-names
drwxrwxr-x  2 lar lar   4096  8月 11 11:14 wildcard
```

cd .., cd .., cd .. ကို သုံးခါရိုက်ပြီး လက်ရှိpath ကနေ အထက်သုံးဆင့်တက်တဲ့ ကိစ္စက နေ့စဉ်လိုလို သုံးရလို့ အောက်ပါအတိုင်း alias လုပ်ထားတော့ အဆင်ပြေတာပေါ့။

In [59]:

```
alias ...='cd ../../'
```

setting လုပ်ထားတဲ့ alias ကို သုံးပြဖို့အတွက်၊ လက်ရှိ ရောက်ရှိနေတဲ့path ကနေ /usr/share/calendar/အောက်ကို အရင်ရွှေ့မယ်။

In [60]:

```
cd /usr/share/calendar/
```

မြင်သာအောင်၊ လက်ရှိရောက်နေတဲ့ path ကို pwd command နဲ့ print ထုတ်ခိုင်းမယ်။

In [61]:

```
pwd
```

```
/usr/share/calendar
```

... command ကိုသုံးပြီး အထက် ၃ဆင့်မှာရှိတဲ့ folderကို ရွှေ့မယ်။

In [62]:

```
...
```

pwd command ရိုက်ကြည့်ပါ။

In [63]:

```
pwd
```

```
/usr
```

လက်ရှိ ဘယ်လို alias တွေရှိသလဲ ဆိုတာကို သိချင်ရင်

In [64]:

```
alias
```

```
alias ...='cd ../../'
alias alert='notify-send --urgency=low -i "$([ $? = 0 ] && echo termin
al || echo error)" "$(history|tail -n1|sed -e '\''s/^\s*[0-9]\+\s*//;
s/[;&|]\s*alert$//'\'')"'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -la'
alias ls='ls --color=auto'
```

12. touch (change file timestamps)

touch command က ဖိုင် တဖိုင်ရဲ့ timestamps (ဖိုင်ကို စဆောက် တဲ့အချိန်၊ ရေးတဲ့ အချိန်၊ ဖတ်တဲ့ အချိန်) တွေကို ပြင်တဲ့အခါသုံးလေ့ရှိတယ်။

ဥပမာ fileA ရဲ့နောက်ဆုံး ပြင်ဆင်ခဲ့တဲ့ အချိန်ကို ls command ကို -l option ပေးပြီး ကြည့်နိုင်တယ်။

In [65]:

```
cd ~/linux-cmd/
ls -l fileA
```

```
-rw-rw-r-- 1 lar lar 67  8月 11 17:48 fileA
```

touch fileA နဲ့ fileA ရဲ့timestamps ကို လက်ရှိအချိန်နဲ့ ပြင်မယ်။
ပြီးရင် ls -l fileA နဲ့ fileA ရဲ့timestamps ကို ပြန်စစ်ကြည့်မယ်။

In [66]:

```
touch fileA
ls -l fileA
```

```
-rw-rw-r-- 1 lar lar 67  8月 11 18:05 fileA
```

"月" ဆိုတဲ့ စာလုံးက ဂျပန်စာမှာ လ ကိုပြောတာပါ။ ဥပမာ ဇွန်လပိုင်းဆိုရင် "6月" ရေးပါတယ်။
နဂိုရှိနေတဲ့ fileA ရဲ့လ၊ ရက် နဲ့ အချိန်တွေ ပြောင်းလဲသွားတာကို တွေ့ရပါလိမ့်မယ်။

Linux OS တွေမှာ POSIX standard သတ်မှတ်ချက်အရ၊ ဖိုင်တဖိုင်ရဲ့ဒေတာတွေကို နောက်ဆုံး ဖတ်တဲ့အချိန် (last data access timestamp)၊ နောက်ဆုံး ဒေတာတွေကို ပြင်ဆင်ခဲ့တဲ့ အချိန် (last data modification timestamp) နဲ့ နောက်ဆုံး ဖိုင်ရဲ့status တွေကို ပြင်ခဲ့တဲ့ အချိန် (last file status change timestamp) ဆိုပြီး ၃မျိုးရှိပါတယ်။ အတိုကောက်အနေနဲ့ last data access timestamp ကို atime၊ last data modification timestamp ကို mtime၊ last file status change timestamp ကို ctime ဆိုပြီးလဲခေါ်ကြပါတယ်။ ဖိုင်တဖိုင်ရဲ့အဲဒီအချိန်တွေကို အသေးစိတ်သိချင်ရင် stat command ကို သုံးပြီးကြည့်နိုင်ပါတယ်။

In [67]:

```
stat fileA
```

```
  File: 'fileA'
  Size: 67              Blocks: 8          IO Block: 4096   regular fi
le
Device: 802h/2050d      Inode: 677359      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/     lar)   Gid: ( 1000/     la
r)
Access: 2017-08-11 18:05:21.191648212 +0900
Modify: 2017-08-11 18:05:21.191648212 +0900
Change: 2017-08-11 18:05:21.191648212 +0900
 Birth: -
```

touch command ရဲ့ -t option ကိုသုံးပြီး ဖိုင်ရဲ့ timestampကို ကိုယ်လိုချင်တဲ့ အချိန်အဖြစ် ပြောင်းခိုင်းလို့ရပါတယ်။ ဖိုင်အသစ်တခုကို လည်း ကိုယ်လိုချင်တဲ့ timestampနဲ့ ပေးပြီး ဆောက်ခိုင်းလို့ရပါတယ်။ ဥပမာ ဖိုင်အသစ်တဖိုင်ကို -t 201502032030.10 option ပေးပြီး touch လုပ်ကြည့်ရအောင်။ ဒီနေရာမှာ အချိန်ကိုပေးရတဲ့ format က [[CC]YY]MMDDhhmm[.ss] ဆိုတဲ့ ပုံစံနဲ့ပါ။ ဒီနေရာမှာ CC ဆိုတာက ခုနှစ်တခုရဲ့ပထမဆုံးစာလုံး၂လုံးကို ဆိုလိုပါတယ်။

In [68]:

```
touch  -t 201502032030.10 newfile
```

In [69]:

```
stat newfile
```

```
  File: 'newfile'
  Size: 0               Blocks: 0          IO Block: 4096   regular em
pty file
Device: 802h/2050d      Inode: 677350      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/     lar)   Gid: ( 1000/     la
r)
Access: 2015-02-03 20:30:10.000000000 +0900
Modify: 2015-02-03 20:30:10.000000000 +0900
Change: 2017-08-11 18:05:23.539648142 +0900
 Birth: -
```

ကျွန်တော် အများဆုံး သုံးဖြစ်တဲ့ ပုံစံကတော့ blank file တခုကို အသစ်လုပ်ချင်ရင် "touch filename" ဆိုပြီး ဖိုင်အသစ်တခုကို ဆောက်တဲ့ ပုံစံပါ။

13. diff (compare files line by line)

ဖိုင်နှစ်ခုကို တူမတူ စာကြောင်းတစ်ကြောင်းချင်းစီ တိုက်စစ်ချင်တဲ့ အခါမှာ သုံးပါတယ်။

အရင်ဆုံး fileA ထဲမှာ ဘာတွေရှိသလဲ ဆိုတာကို cat command နဲ့ ရိုက်ကြည့်မယ်။
-n option ပေးပြီးတော့ လိုင်းနံပါတ်ပါထိုးခိုင်းပါမယ်။ အောက်ပါအတိုင်းမြင်ရပါလိမ့်မယ်။

In [70]:

```
cat -n fileA
```

```
     1  Mingalar bar!
     2  I am fileA.
     3  I was born in April.
     4  My blood type is O.
```

ပြီးတော့ fileB ထဲမှာ ဘာတွေရှိသလဲ ဆိုတာကိုလည်း cat command နဲ့ ရိုက်ကြည့်မယ်။
fileB မှာက ငါးကြောင်းမြောက်မှာ ဘာစာလုံးမှမရှိတဲ့ စာကြောင်းတစ်ကြောင်း ရှိနေကြောင်း တွေ့ရပါလိမ့်မယ်။
ဘာစာမှ မရိုက်ပဲ enter ကီးကို ခေါက်ထားတဲ့ အခါမျိုးမှာ ဒီလိုမျိုး ရှိနေတတ်ပါတယ်။

In [71]:

```
cat -n fileB
```

```
     1  Hi!
     2  I am fileB.
     3  I was born in November.
     4  My blood type is O.
     5
```

အထက်ပါ fileA နဲ့ fileB နှစ်ဖိုင်ကို diff command ကို သုံးပြီး နှိုင်းယှဉ်ကြည့်ရင် အောက်ပါအတိုင်း မြင်ရပါလိမ့်မယ်။

In [72]:

```
diff fileA fileB
```

```
1,3c1,3
< Mingalar bar!
< I am fileA.
< I was born in April.
---
> Hi!
> I am fileB.
> I was born in November.
4a5
>
```

"<" နဲ့ ပြတာက fileA က စာကြောင်း၊ ">" နဲ့ ပြတာက fileB က မတူတဲ့ စာကြောင်းတွေ ပါ။
diff command က command parameter အနေနဲ့ ပထမဆုံးပေးတဲ့ ဖိုင် fileA ကို ဒုတိယ command parameter ဖြစ်တဲ့ fileB နဲ့တူဖို့အတွက် ဘယ်စာကြောင်းတွေကို ပြင်ရမလဲ ဆိုတဲ့ ပုံစံမျိုးနဲ့ ပြပေးပါတယ်။
အသေးစိတ် ရင်းပြရရင်၊ output လုပ်ပေးတဲ့ ပထမဆုံးစာကြောင်း "1,3c1,3" မှာ ပထမ 1,3 ဆိုတာက fileA ရဲ့စာကြောင်းနံပါတ် 1 ကနေ စာကြောင်းနံပါတ် 3 အထိ ကို "c" (change) ပြင်ရမယ်။ ဒုတိယပြတဲ့ 1,3 က fileB ရဲ့စာကြောင်းနံပါတ် 1 ကနေ စာကြောင်းနံပါတ် 3 နဲ့ တူဖို့အတွက်လို ဆိုလိုတာပါ။
"---" က fileA နဲ့ fileB ကို ခွဲခြားပေးထားတဲ့ လိုင်းပါ။
နောက်ဆုံး ပြတဲ့ 4a5 ဆိုတာက ပထမဆုံးဖိုင်က လေးကြောင်းမြောက်နေရာမှာ a (add) စာကြောင်းသစ်အနေနဲ့ ထည့်ရမယ်၊ ဒုတိယဖိုင် fileB က စာကြောင်းနံပါတ် ၅ ကိုလို့ ဆိုလိုပါတယ်။ ဒီနေရာမှာ fileB ရဲ့စာကြောင်း နံပါတ်၅က ဘာမှမရှိဘူးလို့မြင်နေရတဲ့ (Enter တခုပို ခေါက်ထားမိတဲ့) စာကြောင်းပါ။

မှတ်ချက်။ ။ a, b, c ဆိုပြီး ဘယ်လို editing လုပ်ရမယ်ဆိုတာကို အတိုကောက်ပြပြီး အချက်အလက်အနေနဲ့ ဖော်ပြပါတယ်။
ဒီနေရာမှာ a for add၊ c for change၊ d for delete ဆိုပြီး မှတ်သားထားပါ။

diff command ကို -c option ကိုသုံးပြီး contextual mode အနေနဲ့ ပြခိုင်းလိုလဲ ရပါတယ်။

In [73]:

```
diff fileA fileB -c
```

```
*** fileA       2017-08-11 18:05:21.191648212 +0900
--- fileB       2017-08-11 11:14:36.068526803 +0900
***************
*** 1,4 ****
! Mingalar bar!
! I am fileA.
! I was born in April.
  My blood type is O.
--- 1,5 ----
! Hi!
! I am fileB.
! I was born in November.
  My blood type is O.
+
```

အထက်မှာ မြင်ရတာက နှိုင်းယှဉ်ကြည့်နေတဲ့ ဖိုင်နာမည် ၂ခု၊ အဲဒီဖိုင်တွေကို နောက်ဆုံး ပြင်ခဲ့တဲ့ ရက်စွဲ၊ နာရီ ကိုပြတာပါ။
+0900 ဆိုတာက GMT +9၊ ဂျပန် timezone ဖြစ်တဲ့ Greenwich Mean Time +9 ကိုပြောတာပါ။
! အမှတ်အသားက ဖိုင်နှစ်ဖိုင်ကို တူဖို့အတွက် ပြင်ဖို့လိုအပ်တဲ့ စာကြောင်းတွေကို ပြတာပါ။
စာကြောင်းရဲ့ ရှေ့မှာ ဘာမှမပြရင် တူလို့ပါ (အထက်ကဥပမာ အရဆိုရင် My blood type is O. ဆိုတဲ့ စာကြောင်းပါ)။
+ အမှတ်အသားက နှိုင်းယှဉ်တဲ့ ဖိုင်နှစ်ဖိုင်အနက်က ဒုတိယဖိုင်မှာပဲ ရှိတဲ့စာကြောင်းဖြစ်ပြီး၊ fileA မှာ ဝင်ဖြည့်ရမယ့် စာကြောင်းကို ပြတာပါ။

diff command ကို -u option ကိုသုံးပြီး unify mode အနေနဲ့လဲ ပြခိုင်းလို့ ရပါတယ်။

In [74]:

```
diff fileA fileB -u
```

```
--- fileA       2017-08-11 18:05:21.191648212 +0900
+++ fileB       2017-08-11 11:14:36.068526803 +0900
@@ -1,4 +1,5 @@
-Mingalar bar!
-I am fileA.
-I was born in April.
+Hi!
+I am fileB.
+I was born in November.
 My blood type is O.
+
```

-u option က အကြမ်းမျဉ်းအားဖြင့်တော့ context mode နဲ့ ဆင်တူပါတယ်။
တူတဲ့ စာကြောင်းတွေကို နှစ်ခါမပြတာ နဲ့ ဖိုင်နှစ်ဖိုင် အနေနဲ့ သပ်သပ်စီခွဲမပြတာပဲ၊ ပေါင်းပြီး ပြတာပဲ ကွာပါတယ်။

diff command ကို -y option ကိုသုံးပြီး ဖိုင်နှစ်ဖိုင်ကို စာကြောင်းတကြောင်းချင်းစီကို ဘေးချင်းယှဉ်ပြီး (သို့) ကော်လံ နှစ်ခု အနေနဲ့ ပြခိုင်းတာမျိုးလဲ လုပ်နိုင်ပါတယ်။
လူအနေက ပုံမှန်အားဖြင့် စာကြောင်းနှစ်ကြောင်းကို ဘေးချင်းစီပြီးကြည့်တာက ပိုအဆင်ပြေမယ်လို့ ထင်ပါတယ်။

In [75]:

```
diff fileA fileB -y
```

```
Mingalar bar!                                                   | Hi!
I am fileA.                                                     | I am f
ileB.
I was born in April.                                            | I was
 born in November.
My blood type is O.                                               My blo
od type is O.
                                                                >
```

ဒီနေရာမှာ သုံးထားတဲ့ သင်္ကေတတွေရဲ့အဓိပ္ပါယ်တွေကတော့
"|" က ဖိုင်နှစ်ဖိုင်မှာ မတူတဲ့ စာကြောင်းတွေကို ပြတာပါ။
">" က ညာဖက်အခြမ်း (သို့) ဒုတိယဖိုင်ဖြစ်တဲ့ fileB ဘက်မှာပဲ ရှိတဲ့ စာကြောင်းကို ပြတာပါ။
(ဒီနေရာမှာ မျက်လုံးနဲ့ မမြင်နိုင်တဲ့ Enter ခေါက်ထားလို့ပါ)
ဘာ သင်္ကေတမှ မပြရင် စာကြောင်းနှစ်ကြောင်းက တူနေလို့ပါ။
အထက်က ဥပမာမှာ မဆိုင်လို့ မပြထားပေမဲ့၊ နောက်ထပ် သင်္ကေတ တစ်ခု ကျန်ပါသေးတယ်။ အဲဒါက "<" သင်္ကေတပါ။ "<" ကတော့ ဘယ်ဘက်အခြမ်းကဖိုင် (သို့) နှိုင်းယှဉ်ကြည့်နေတဲ့ အထဲက ပထမဖိုင်မှာပဲ ရှိတဲ့ စာကြောင်းမျိုးအတွက် သုံးပါတယ်။

မှတ်ချက်။ ။ အထက်က ဥပမာ "diff fileA fileB -y" ကို sdiff command ကိုသုံးပြီး "sdiff fileA fileB" ဆိုပြီးလည်း run နိုင်ပါတယ်။

diff command ကို နောက်ဆုံး ဥပမာအနေနဲ့ ဖိုင်တွေကို နှိုင်းယှဉ်ကြည်တာတင် မကပဲ၊ ဖိုလ်ဒါ နှစ်ခုကိုလည်း နှိုင်းယှဉ်လို့ရကြောင်း ပြပါမယ်။

အရင်ဆုံး လက်ရှိ ဖိုလ်ဒါအောက်မှာက ဘာတွေရှိသလဲ ဆိုတာကို ls နဲ့ ကြည့်မယ်။
ပြီးရင် ဖိုလ်ဒါအသစ်တခုဆောက်ပြီး၊ ဖိုင်တချို့ကို ကူးထည့်ပါမယ်။

In [76]:

```
ls -F
```

```
all-vi-info.txt  fileA          linux-commands.ipynb      otest.tag
data1/           fileB          linux-commands.ipynb.bak  otest.word
data2/           fmt.out        ls.out                    screen/
data3/           folderA/       my-news                   sorted-names
data4/           folder-athit/  newfile                   wildcard/
data5/           folder-new/    news
echo             item-list      otest
```

လောလောဆယ် လက်ရှိ path အောက်မှာ folderA/ ပဲရှိတယ်။
mkdir command နဲ့ folderB/ ကိုဆောက်မယ်။

In [77]:

```
mkdir folderB
```

ဆောက်ထားတဲ့ folderB/ အောက်ကို ဖိုင်တချို့ကော်ပီကူးထည့်မယ်။

In [78]:

```
cp fileA ./folderB/
```

In [79]:

```
cp otest ./folderB/
```

In [80]:

```
cp ./folderA/127-135.pdf ./folderB/
```

အထက်မှာ ပြခဲ့တဲ့အတိုင်း လက်ရှိ path အောက်မှာရှိနေတဲ့ fileA ဆိုတဲ့ ဖိုင်နဲ့ otest ဆိုတဲ့ ဖိုင် ကို folderB/ အောက်ကို ကော်ပီကူးထည့်ခဲ့ပါတယ်။
အဲဒီနောက်မှာ လက်ရှိ path ရဲ့အောက်က folderA/ ဖိုလ်ဒါအောက်မှာရှိတဲ့ 127-135.pdf ဆိုတဲ့ ဖိုင်ကိုလည်း folderB/ အောက်ကို ကော်ပီကူးထည့်ခဲ့ပါတယ်။
အဲဒါကြောင့် folderB/ အောက်မှာ ဖိုင် ၃ဖိုင် ရှိနေပါပြီ။
ls command နဲ့ folderB/ နဲ့ folderA/ အောက်မှာရှိတဲ့ ဖိုင်တွေကို ကြိုကြည့်ထားရအောင်။

In [ ]:

```
ls ./folderB/
```

In [82]:

```
ls ./folderA/
```

```
127-135.pdf
```

diff command ကိုသုံးပြီး folderA/ နဲ့ folderB/ ကိုနှိုင်းယှဉ်ကြည့်ရအောင်။

In [83]:

```
diff folderA/ folderB/
```

```
Only in folderB/: fileA
Only in folderB/: otest
```

ဖိုလ်ဒါ နှစ်ခုကို နှိုင်းယှဉ်ကြည့်ရတာက၊ ဖိုင်နှစ်ခုကို နှိုင်းယှဉ်ကြည့်ရတာထက်စာရင် output ကပိုပြီး user friendly ဖြစ်ပါတယ်။
folderA/ ထဲမှာပဲ ရှိတဲ့ ဖိုင်တွေ၊ ဖိုလ်ဒါတွေကို Only in folderA/ ဆိုပြီး ပြပါတယ်။
ထိုနည်းလည်းကောင်းပဲ folderB/ ထဲမှာပဲ ရှိတဲ့ ဖိုင်တွေ၊ ဖိုလ်ဒါတွေကိုလည်း Only in folderB/ ဆိုပြီး ပြပေးပါတယ်။

14. less (opposite of more)

cat command နဲ့ ဖိုင်တွေကို ဖတ်ကြည့်လို့ရပေမဲ့၊ ဖိုင်ထဲမှာရှိတဲ့ စာကြောင်းအရေအတွက်က မော်နီတာစကရင် တခုစာထက်မက များနေတဲ့ အခါမျိုးမှာ less command ကို သုံးပါ။ text file တွေထဲမှာရှိတဲ့ စာကြောင်းတွေကို စာမျက်နှာ တစ်မျက်နှာချင်း အပေါ်တက်၊ အောက်ဆင်း (Page-Up, Page-Down) လုပ်ပြီး ကြည့်လို့ရအောင် လုပ်ပေးတဲ့ command ပါ။

သုံးပုံသုံးနည်းက အရမ်းလွယ်ပါတယ်။ less command ရဲ့နောက်မှာ ကိုယ်ကြည့်ချင်တဲ့ ဖိုင်နာမည်ကို ပေးလိုက်ယုံပါပဲ။

In [84]:

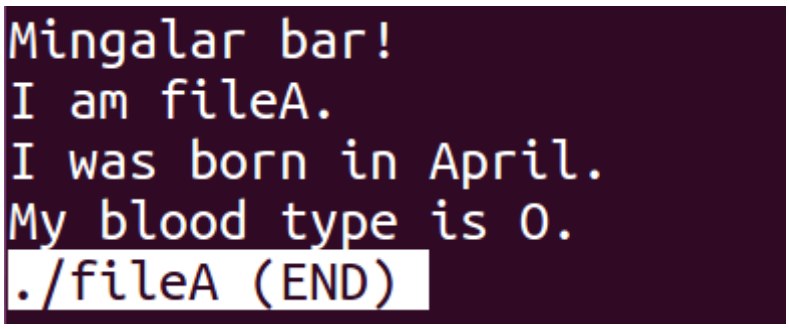
```
ls
```

```
all-vi-info.txt  fileA         item-list                 otest
data1            fileB         linux-commands.ipynb      otest.tag
data2            fmt.out       linux-commands.ipynb.bak  otest.word
data3            folderA       ls.out                    screen
data4            folder-athit  my-news                   sorted-names
data5            folderB       newfile                   wildcard
echo             folder-new    news
```

In [ ]:

```
less ./fileA
```

မော်နီတာမှာမြင်ရမဲ့ text editor လိုမျိုး output ကို ပုံအနေနဲ့ ထည့်ပြထားပါတယ်။



ဒီနေရာမှာက ဥပမာအနေနဲ့ စာကြောင်းရေ လေးကြောင်းသာရှိတဲ့ fileA ကို less နဲ့ run ပြထားပေမဲ့၊ လက်တွေ့ စမ်းကြည့်တဲ့ အခါ စာကြောင်းရေ အများကြီးရှိတဲ့ ဖိုင်နဲ့ စမ်းသပ်ကြည့်ပါ။ စာကြောင်းရေ အများကြီးရှိတဲ့ ဖိုင်နဲ့ ဆိုရင် စကရင်ရဲ့အောက်ဆုံး ဘယ်ဘက်ထောင့်မှာ ":" က ပေါ်နေပြီး၊ command အမျိုးမျိုးသုံးပြီး ဖိုင်ကို စာမျက်နှာတစ်ခုချင်း ရှေ့တိုး၊ နောက်ဆုတ် လုပ်တာ၊ ရှာချင်တဲ့ စာလုံးကို "/" (သို့) "?" ရိုက်ပြီး Regular Expression pattern တွေနဲ့ ရှာတာဖွေတာ စသည်ဖြင့် အလုပ်အများကြီးလုပ်ပေးလို့ less command က linux terminal မှာ text file တွေကို view လုပ်ကြည့်တဲ့ နေရာမမှာ မသိမဖြစ်နဲ့ သုံးသင့်တဲ့ command တစ်ခုဖြစ်ပါတယ်။ less command ကို run နေစဉ်မှာ ဘယ်လို command တွေကိုသုံးပြီး ရှေ့တိုး၊ နောက်ဆုတ်၊ ရှာဖွေ စတဲ့ အလုပ်တွေကို လုပ်လို့ရသလဲ ဆိုတာကို သိချင်ရင်၊ "h" ကို နှိပ်ပြီး Help Screen ကို ခေါ်ကြည့်နိုင်ပါတယ်။

less command နဲ့ ဖိုင်ကိုကြည့်နေရာကနေ ထွက်ချင်ရင် "q" (သို့) "Q" ကီးကိုနှိပ်ပြီး ထွက်နိုင်ပါတယ်။

အသုံးများတဲ့ option တွေကတော့ အောက်ပါအတိုင်း ဖြစ်ပါတယ်။

less -N filename
-N က ဖိုင်ထဲက စာကြောင်း တစ်ကြောင်းချင်းစီကို လိုင်းနံပါတ် တပ်ပေးပြီး ပြစေချင်တဲ့ အခါ အသုံးပြု

less -N +100 filename
+ နဲ့ ကိုယ်စပြစေချင်တဲ့ စာကြောင်းနံပါတ်ကို ပေးလို့ရတယ်။
အထက်က ဥပမာက စာကြောင်းနံပါတ် 100 ကနေ စပြပေးပါလို့ command ပေးတာ။

less +F experiment1.log

+F က log file လိုမျိုး အပြောင်းအလဲက စဉ်ဆက်မပြတ်ရှိနေတဲ့ ဖိုင်တွေရဲ့နောက်ဆုံးစာမျက်နှာ (updated page) တွေကို စောင့်ကြည့်ချင်တဲ့ အခါမျိုးမှာ သုံးပါတယ်။

ဥပမာ ကျွန်တော်တို့က experiment1 ဆိုပြီး စမ်းသပ်မှုတစ်ခုလုပ်နေပြီး၊ အဲဒီကနေ ထွက်လာတဲ့ ရလဒ်တွေ၊ error တွေကို experiment1.log ဖိုင်မှာ ဝင်ပြီးရေးပေးနေတယ်ဆိုပါစို့။ အဲဒီ စမ်းသပ်မှုကနေ ဘယ်လို log တွေဝင်ရေးသလဲ ဆိုတာကို less +F ကို သုံးပြီး ကြည့်တာမျိုးလုပ်ပါတယ်။ experiment1 စမ်းသပ်မှုမှာပါဝင်တဲ့ ပရိဂရမ်က experiment1.log ဖိုင်မှာ တခုခုဝင်ရေးတာ၊ ဖျက်တာကို

လုပ်တာနဲ့ တပြိုင်နက်၊ အဲဒီအပြောင်းအလဲတွေကို ကြည့်နေတဲ့ less editor ရဲ့စကရင်မှာ active ဖြစ်လို့ပါ။

ဒါအပြင်၊ လက်ရှိ F mode ကနေ Ctrl + C နဲ့ ခဏထွက်ပြီး ပုံမှန် less နဲ့ ကြည့်နေတဲ့ mode ဆီကို ပြောင်းပြီး ကိုယ်ရာကြည့်ချင်တဲ့ စာလုံးကို ရှာ၊ စာကြောင်းဆီကို သွား စတာတွေလုပ်လို့ ရပါတယ်။ ပြီးရင် F ကို နှိပ်ပြီး၊ F mode ဆီကိုပြန်သွားပြီး log ဖိုင်ရဲ့နောက်ဆုံးအပြောင်းလဲ ကို ဆက်စောင့်ကြည့်နိုင်ပါတယ်။

မှတ်ချက်။ ။ ခေါင်းစဉ်မှာ opposite of more ဆိုပြီး ရေးထားတာက၊ less က more ဆိုတဲ့ command နဲ့ ဆန့်ကျင်ဘက်ပါ။ more command ထက် အလုပ်အများကြီး လုပ်ပေးနိုင်ပါတယ်လို့ ဆိုလိုပါတယ်။ man less ဆိုပြီး ရိုက်ကြည့်ရင် ရှင်းပြထားတာကို ဖတ်လို့ရပါတယ်။

15. type (display a command type)

type command က command တစ်ခုချင်းစီရဲ့အမျိုးအစားကို သိချင်တဲ့ အခါမှာသုံးပါတယ်။ command တစ်ခုခုရဲ့နာမည်ကို type ကို သုံးပြီး အဲဒီ command က alias လား၊ shell ရဲ့function လား၊ shell ရဲ့builtin လား၊ disk ထဲမှာသိမ်းထားတဲ့ file လား၊ shell ရဲ့ reserved word လား ဆိုတာကို ရှာဖွေနိုင်ပါတယ်။

ဥပမာ type ls ဆိုပြီး ရိုက်ကြည့်ရင် အောက်ပါအတိုင်း ပြပေးပါလိမ့်မယ်။

ls is aliased to `ls --color=auto'

ls command က "ls --color=auto" ဆိုတဲ့ command ကို alias လုပ်ထားတာဖြစ်ကြောင်းကို သိနိုင်ပါတယ်။

In [10]:

```
type cd
```

```
cd is a shell builtin
```

cd command က shell ရဲ့builtin command ဖြစ်ကြောင်းကို ပြပေးပါတယ်။

-t option ပေးပြီး command တစ်ခုရဲ့အမျိုးအစားကို စာလုံးတစ်လုံးနဲ့ တိုတိုပြခိုင်းလို့ ရပါတယ်။ alias လုပ်ထားတာဆိုရင် "alias"၊ shell ရဲ့reserved word ဆိုရင် "keyword" ဆိုပြီး၊ shell ရဲ့function ဆိုရင် "function" ဆိုပြီး၊ shell ရဲ့builtin command ဆိုရင် "builtin"၊ disk မှာ သိမ်းထားတဲ့ ဖိုင်တစ်ဖိုင် ဆိုရင် "file" ဆိုပြီး အသီးသီးပြပေးပါလိမ့်မယ်။

အောက်ဖော်ပြပါ type command ဥပမာ တစ်ချို့ကို terminal မှာ ရိုက်စမ်းကြည့်ပါ။

In [11]:

```
type -t if
```

```
keyword
```

if က bash shell ရဲ့keyword ဖြစ်ကြောင်းကို ပြပေးပါလိမ့်မယ်။

In [12]:

```
type -t rm
```

```
file
```

"rm command" က file ဖြစ်ကြောင်း ပြသပေးပါလိမ့်မယ်။

In [13]:

```
type -t pwd
```

```
builtin
```

builtin ဆိုပြီး ပြသပေးပါလိမ့်မယ်။

command prompt မှာလဲ function တစ်ခုကို ဆောက်လို့ရပါတယ်။ ဥပမာ အနေနဲ့ mcd ဆိုတဲ့function တစ်ခုကို ဆောက်ပြပါမယ်။
mcd က mkdir နဲ့ ဖိုလ်ဒါအသစ်တစ်ခုကို ဆောက်ပြီးရင်၊ အဲဒီ ဆောက်လိုက်တဲ့ ဖိုလ်ဒါအသစ်ထဲကို တခါတည်းဝင်ပေးမဲ့ function ပါ။

In [14]:

```
mcd () { mkdir -p $1; cd $1; }
```

ဒီနေရာမှာ $1 ဆိုတာက mcd function ကိုခေါ်သုံးတဲ့ အခါ နောက်က ပေးမဲ့ ဖိုလဒ်ဒါနာမည် (သို့) command line parameter ကို ဆိုလိုပါတယ်။
ဆောက်ပြီးသွားရင် အောက်ပါအတိုင်း mcd function ကို folder-new ဆိုတဲ parameter နဲ့ run ခိုင်းပြီး၊ လက်ရှိရောက်နေတဲ့ path ကို pwd command နဲ့ ပြခိုင်းပါမယ်။
ဒီနေရာမှာ ";" က command သုံးလေးခု ဆက်တိုက်ကို ရေးတဲ့ အခါမှာ၊ command တစ်ခုပြီးဆုံးကြောင်းအတွက် သုံးတဲ့ symbol ပါ။

In [15]:

```
mcd folder-new; pwd;
```

```
/home/lar/linux-cmd/folder-new/folder-new
```

/home/lar/linux-cmd/folder-new ဆိုပြီး ရိုက်ပြပေးတာက pwd ရဲ့ output ဖြစ်တဲ့ လက်ရှိရောက်နေတဲ့ path ပါ။
mcd အမိန့်က တကယ် အလုပ်လုပ်ပြီးသွားသလား ဆိုတာကို သေချာအောင် ရိုက်ခိုင်းထားတာပါ။

mcd ကို type -t option နဲ့ ရိုက်ကြည့်ရအောင်။

In [16]:

```
type -t mcd
```

```
function
```

အထက်ပါအတိုင်း mcd က function ဖြစ်ကြောင်းကို type command က ပြပေးနိုင်ပါတယ်။

command က path တစ်ခုအောက်မှာထက်ပိုရှိလား သိချင်ရင် -a option နဲ့ ကြည့်နိုင်ပါတယ်။

In [17]:

```
type -a python
```

```
python is /home/lar/anaconda3/bin/python
python is /home/lar/anaconda3/bin/python
python is /home/lar/anaconda3/bin/python
python is /usr/bin/python
```

"-P" option နဲ့ ပေးလိုက်တဲ့ command ရဲ့path ကို ရာခိုင်းလို့ရပါတယ်။
ဥပမာ "touch" command ရဲ့ပရိုဂရမ်ဖိုင်က ဘယ်မှာရှိသလဲဆိုတာကို သိချင်ရင် အောက်ပါအတိုင်း "type -P touch" ဆိုပြီး command

ပေးပါ။

In [9]:

```
type -P touch
```

```
/usr/bin/touch
```

"cd" command ရဲ့path ကို "-P" option နဲ့ရှာခိုင်းရင်တော့ ဘာမှရိုက်ပြမှာ မဟုတ်ပါဘူး။

In [10]:

```
type -P cd
```

ဘာကြောင့်လဲ ဆိုတော့ "cd" command က shell ရဲ့builtin command ဖြစ်နေလို့ပါ။ အထက်မှာသင်ပေးခဲ့သလို "touch cd" နဲ့ ရိုက်ကြည့်ရင် "cd is a shell builtin" ဖြစ်ကြောင်းရိုက်ပြပေးပါလိမ့်မယ်။ ဒီတခါတော့ "-t" option ကို သုံးပြီး စာလုံးတစ်လုံးထဲနဲ့ပဲ ရိုက်ပြခိုင်းပြီး confirm လုပ်ရအောင်။

In [17]:

```
type -t cd
```

```
builtin
```

command prompt မှာ အကြိမ်ပေါင်းများစွာ ရိုက်သုံးရတဲ့ "ls" ရဲ့path ကိုလည်း ရှာကြည့်ရအောင်။

In [16]:

```
type -P ls
```

```
/bin/ls
```

16. which (locate a command)

which နဲ့ command တစ်ခုရဲ့executable လုပ်မဲ့ path ကို ရှာကြည့်လို့ရပါတယ်။ command တစ်ခုက version တစ်ခုထက်မက ပို့ရှိတဲ့ အခြေအနေမျိုးမှာဆိုရင်လည်း ဘယ် version (သို့) ဘယ် path အောက်က command ကို ခေါ် run မှာလဲဆိုတာကို သိချင်တဲ့ အခါမှာ အသုံးဝင်ပါတယ်။

ဥပမာ firefox ကို ဘယ် path ကနေ ခေါ် run မှာလဲ ဆိုတာကို which နဲ့ ကြည့်ရအောင်

In [18]:

```
which firefox
```

```
/usr/bin/firefox
```

တစ်ခုသိထားရမှာက which က shell ရဲ့builtin command တွေနဲ့ alias တွေကို ပြပေးမှာ မဟုတ်ဘူးဆိုတာကိုပါ။

In [19]:

```
which cd
```

အထက်မှာ ဥပမာပြထားခဲ့သလို shell ရဲ့builtin command တစ်ခုဖြစ်တဲ့ cd ကို which နဲ့ ကြည့်ရင် ဘာမှပြပေးမှာ မဟုတ်ပါ။

17. --help or -h (display usage information)

--help (သို့) -h က command တစ်ခုကို ဘယ်လို run ရမလဲဆိုတာ၊ ဘာ option တွေရှိသလဲ ဆိုတာကို သိချင်တဲ့ အခါမှာ သုံးပါတယ်။ ဘယ်လို command မျိုးမှာမဆို --help option ကပါဝင်ပြီး၊ command line interface (CLI) မှာတော့ သိကို သိထားရပါမယ်။ သုံးပုံ သုံးနည်း ဥပမာတွေကတော့ အောက်ပါအတိုင်းပါပဲ။

mkdir command ရဲ့ help ကိုကြည့်ချင်တဲ့ အခါ

In [20]:

```
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 to
o.
  -m, --mode=MODE   set file mode (as in chmod), not a=rwx - umask
  -p, --parents     no error if existing, make parent directories as n
eeded
  -v, --verbose     print a message for each created directory
  -Z                   set SELinux security context of each created di
rectory
                         to the default type
      --context[=CTX]  like -Z, or if CTX is specified then set the SE
Linux
                         or SMACK security context to CTX
      --help     display this help and exit
      --version  output version information and exit

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/mkdir>
or available locally via: info '(coreutils) mkdir invocation'
```

diff command ရဲ့ help ကို ကြည့်ချင်တဲ့ အခါမှာ

In [21]:

```
diff --help
```

```
Usage: diff [OPTION]... FILES
Compare FILES line by line.

Mandatory arguments to long options are mandatory for short options to
o.
      --normal                  output a normal diff (the default)
  -q, --brief                   report only when files differ
  -s, --report-identical-files  report when two files are the same
  -c, -C NUM, --context[=NUM]   output NUM (default 3) lines of copied
context
  -u, -U NUM, --unified[=NUM]   output NUM (default 3) lines of unifie
d context
  -e, --ed                      output an ed script
  -n, --rcs                     output an RCS format diff
  -y, --side-by-side            output in two columns
  -W, --width=NUM               output at most NUM (default 130) print
columns
      --left-column             output only the left column of common
 lines
      --suppress-common-lines   do not output common lines

  -p, --show-c-function         show which C function each change is i
n
  -F, --show-function-line=RE   show the most recent line matching RE
      --label LABEL             use LABEL instead of file name
                                  (can be repeated)

  -t, --expand-tabs             expand tabs to spaces in output
  -T, --initial-tab             make tabs line up by prepending a tab
      --tabsize=NUM             tab stops every NUM (default 8) print
 columns
      --suppress-blank-empty    suppress space or tab before empty out
put lines
  -l, --paginate                pass output through 'pr' to paginate i
t

  -r, --recursive                 recursively compare any subdirectori
es found
      --no-dereference            don't follow symbolic links
  -N, --new-file                  treat absent files as empty
      --unidirectional-new-file   treat absent first files as empty
      --ignore-file-name-case     ignore case when comparing file name
s
      --no-ignore-file-name-case  consider case when comparing file na
mes
  -x, --exclude=PAT               exclude files that match PAT
  -X, --exclude-from=FILE         exclude files that match any pattern
in FILE
  -S, --starting-file=FILE        start with FILE when comparing direc
tories
      --from-file=FILE1           compare FILE1 to all operands;
                                    FILE1 can be a directory
      --to-file=FILE2             compare all operands to FILE2;
                                    FILE2 can be a directory

  -i, --ignore-case               ignore case differences in file cont
ents
```

```
    -E, --ignore-tab-expansion      ignore changes due to tab expansion

    -Z, --ignore-trailing-space     ignore white space at line end
    -b, --ignore-space-change       ignore changes in the amount of whit
  e space
    -w, --ignore-all-space          ignore all white space
    -B, --ignore-blank-lines        ignore changes where lines are all b
  lank
    -I, --ignore-matching-lines=RE  ignore changes where all lines match
  RE

    -a, --text                      treat all files as text
        --strip-trailing-cr         strip trailing carriage return on in
  put

    -D, --ifdef=NAME                output merged file with '#ifdef NAM
  E' diffs
        --GTYPE-group-format=GFMT   format GTYPE input groups with GFMT
        --line-format=LFMT          format all input lines with LFMT
        --LTYPE-line-format=LFMT    format LTYPE input lines with LFMT
      These format options provide fine-grained control over the output
        of diff, generalizing -D/--ifdef.
      LTYPE is 'old', 'new', or 'unchanged'.  GTYPE is LTYPE or 'change
  d'.
      GFMT (only) may contain:
        %<  lines from FILE1
        %>  lines from FILE2
        %=  lines common to FILE1 and FILE2
        %[-][WIDTH][.[PREC]]{doxX}LETTER  printf-style spec for LETTER
          LETTERs are as follows for new group, lower case for old grou
  p:
             F  first line number
             L  last line number
             N  number of lines = L-F+1
             E  F-1
             M  L+1
        %(A=B?T:E)  if A equals B then T else E
      LFMT (only) may contain:
        %L  contents of line
        %l  contents of line, excluding any trailing newline
        %[-][WIDTH][.[PREC]]{doxX}n  printf-style spec for input line nu
  mber
      Both GFMT and LFMT may contain:
        %%  %
        %c'C'  the single character C
        %c'\OOO'  the character with octal code OOO
        C    the character C (other characters represent themselves)

    -d, --minimal            try hard to find a smaller set of changes
        --horizon-lines=NUM  keep NUM lines of the common prefix and suf
  fix
        --speed-large-files  assume large files and many scattered small
  changes

        --help               display this help and exit
    -v, --version            output version information and exit

  FILES are 'FILE1 FILE2' or 'DIR1 DIR2' or 'DIR FILE...' or 'FILE... DI
  R'.
  If --from-file or --to-file is given, there are no restrictions on FIL
  E(s).
  If a FILE is '-', read standard input.
```

```
Exit status is 0 if inputs are the same, 1 if different, 2 if trouble.

Report bugs to: bug-diffutils@gnu.org
GNU diffutils home page: <http://www.gnu.org/software/diffutils/>
General help using GNU software: <http://www.gnu.org/gethelp/>
```

18. man (system's manual pager)

man က executable ပရိဂရမ်တွေ ရဲ့သုံးပုံသုံးနည်း အသေးစိတ်ကို ရှာဖွေကြည့်ချင်တဲ့ အခါမှာ သုံးပါတယ်။ command တွေရဲ့ reference manual ပါပဲ။

--help နဲ့ ကြည့်တာထက် ပိုပြီးပြည့်စုံပါတယ်။ သုံးပုံသုံးနည်းက man program-name ဆိုတဲ့ပုံစံပါ။
ဥပမာ ls command ရဲ့ man page ကို ဖတ်ချင်တယ်ဆိုရင်

man ls

Linux system အများစုမှာ man စာမျက်နှာတွေကို less နဲ့ ပြပေးပါတယ်။ အဲဒါကြောင့် less command မှာ ကြည့်ရင်းနဲ့ သုံးတဲ့ command တွေ (ဥပမာ f နဲ့ စာမျက်နှာတစ်မျက်နှာစာ ရှေ့တိုးတာ၊ b နဲ့ စာမျက်နှာ တစ်မျက်နှာစာ နောက်ဆုတ်တာ၊ ! နဲ့ less ထဲကနေ မထွက်ပဲ command တွေကို run တာမျိုး) အားလုံးကို အသုံးပြုနိုင်ပါတယ်။

help ကြည့်ချင်တဲ့ command ရဲ့နာမည်ကို သေသေချာချာ မသိတဲ့ အခါ၊ နောက်ပြီး ရိုက်ထည့်လိုက်တဲ့ စာလုံးပါတဲ့ (သို့) ဆက်စပ်မှုရှိတဲ့ man စာမျက်နှာတွေကို ရှာကြည့်ချင်တဲ့ အခါမှာ -k option ကိုသုံးပါတယ်။ ဥပမာ python ဆိုတဲ့စာလုံးပါတဲ့ man စာမျက်နှာတွေအားလုံးကို အောက်ပါအတိုင်း ရှာကြည့်နိုင်ပါတယ်။

In [22]:

```
man -k python
```

```
2to3 (1)             - Python2 to Python3 converter
2to3-2.7 (1)         - Python2 to Python3 converter
2to3-3.5 (1)         - Python2 to Python3 converter
dh_python2 (1)       - calculates Python dependencies, adds maintainer
script...
dh_python3 (1)       - calculates Python dependencies, adds maintainer
script...
jwt3 (1)             - Python implementation of JSON Web Token
pdb (1)              - the Python debugger
pdb2.7 (1)           - the Python debugger
pdb3 (1)             - the Python debugger
pdb3.5 (1)           - the Python debugger
py3compile (1)       - byte compile Python 3 source files
py3versions (1)      - print python3 version information
pybuild (1)          - invokes various build systems for requested Pyt
hon ver...
pycompile (1)        - byte compile Python source files
pydoc (1)            - the Python documentation tool
pydoc2.7 (1)         - the Python documentation tool
pydoc3 (1)           - the Python documentation tool
pydoc3.5 (1)         - the Python documentation tool
pygettext (1)        - Python equivalent of xgettext(1)
pygettext2.7 (1)     - Python equivalent of xgettext(1)
pygettext3 (1)       - Python equivalent of xgettext(1)
pygettext3.5 (1)     - Python equivalent of xgettext(1)
python (1)           - an interpreted, interactive, object-oriented pr
ogrammi...
python-config (1)    - output build options for python C/C++ extension
s or em...
python2 (1)          - an interpreted, interactive, object-oriented pr
ogrammi...
python2-config (1)   - output build options for python C/C++ extension
s or em...
python2.7 (1)        - an interpreted, interactive, object-oriented pr
ogrammi...
python2.7-config (1) - output build options for python C/C++ extension
s or em...
python3 (1)          - an interpreted, interactive, object-oriented pr
ogrammi...
python3.5 (1)        - an interpreted, interactive, object-oriented pr
ogrammi...
python3.5m (1)       - an interpreted, interactive, object-oriented pr
ogrammi...
python3m (1)         - an interpreted, interactive, object-oriented pr
ogrammi...
pyversions (1)       - print python version information
x86_64-linux-gnu-python-config (1) - output build options for python
 C/C++ ex...
x86_64-linux-gnu-python2.7-config (1) - output build options for pytho
n C/C++...
```

man help စာမျက်နှာတွေက command အမျိုးအစားပေါ် မူတည်ပြီး၊ သက်ဆိုင်ရာ section တွေ ခွဲပြီးရင်းပြထားတာတွေရှိပါတယ်။ section တွေကို အောက်ပါအတိုင်း နံပါတ်တွေခွဲထားပါတယ်။

1 Executable programs or shell commands
2 System calls (functions provided by the kernel)
3 Library calls (functions within program libraries)
4 Special files (usually found in /dev)
5 File formats and conventions eg /etc/passwd
6 Games
7 Miscellaneous (including macro packages and conventions), e.g. man(7), groff(7)
8 System administration commands (usually only for root)
9 Kernel routines [Non standard]

command တစ်ခု ရဲ့man help စာမျက်နှာတွေ အများကြီးထဲကမှ၊ ကိုယ်ကြည့်ချင်တဲ့ section ကို နံပါတ်နဲ့ option ပေးပြီး ကြည့်လို့ရပါတယ်။
ဥပမာ /etc/passwd ရဲ့ဖိုင်က ဘယ်လို format လဲဆိုတာကို သိချင်ရင် အောက်ပါအတိုင်း ရိုက်ကြည့်ပါ။

In [23]:

```
man 5 passwd
```

```
PASSWD(5)                File Formats and Conversions                P
ASSWD(5)

NAME
       passwd - the password file

DESCRIPTION
       /etc/passwd contains one line for each user account, with seven
fields
       delimited by colons (":"). These fields are:

       ·   login name

       ·   optional encrypted password

       ·   numerical user ID

       ·   numerical group ID

       ·   user name or comment field

       ·   user home directory

       ·   optional user command interpreter

       The encrypted password field may be blank, in which case no pas
sword is
       required to authenticate as the specified login name. However,
 some
       applications which read the /etc/passwd file may decide not to
 permit
       any access at all if the password field is blank. If the passwo
rd field
       is a lower-case "x", then the encrypted password is actually st
ored in
       the shadow(5) file instead; there must be a corresponding line
 in the
       /etc/shadow file, or else the user account is invalid. If the p
assword
       field is any other string, then it will be treated as an encryp
ted
       password, as specified by crypt(3).

       The comment field is used by various system utilities, such as
       finger(1).

       The home directory field provides the name of the initial worki
ng
       directory. The login program uses this information to set the v
alue of
       the $HOME environmental variable.

       The command interpreter field provides the name of the user's c
ommand
       language interpreter, or the name of the initial program to exe
cute.
       The login program uses this information to set the value of the
```

```
$SHELL

       environmental variable. If this field is empty, it defaults to
 the
       value /bin/sh.

FILES
       /etc/passwd
           User account information.

       /etc/shadow
           optional encrypted password file

       /etc/passwd-
           Backup file for /etc/passwd.

           Note that this file is used by the tools of the shadow tool
suite,
           but not by all user and password management tools.

SEE ALSO
       crypt(3), getent(1), getpwnam(3), login(1), passwd(1), pwck(8),
       pwconv(8), pwunconv(8), shadow(5), su(1), sulogin(8).

shadow-utils 4.2                  05/16/2017                          P
ASSWD(5)
```

19. whatis (display one-line manual page descriptions)

command တစ်ခုက ဘာလုပ်ဖို့အတွက်သုံးတာလဲ ဆိုတာကို စာကြောင်းတို တစ်ကြောင်းနဲ့ ရှင်းပြထားတာကို ကြည့်ဖို့အတွက်ဆိုရင် whatis ကို သုံးပါတယ်။ တခါတလေ မှာ command တစ်ခုရဲ့ help စာမျက်နှာတွေကို အသေးစိတ် မဖတ်ချင်ဘူး။ ဘာလုပ်တဲ့ command ဆိုတာကိုပဲ confirm လုပ်ချင်တဲ့ အခါမျိုးမှာ အသုံးများပါတယ်။

ဥပမာ ping ဆိုတဲ့ command က ဘာလုပ်တာလဲ ဆိုတာကို သိချင်တဲ့ အခါ

In [24]:

```
whatis ping
```

```
ping (8)              - send ICMP ECHO_REQUEST to network hosts
```

နောက်ထပ် ဥပမာ တစ်ခုအနေနဲ့ ssh command က ဘာလုပ်ဖို့အတွက်သုံးတာလဲ ဆိုတာကို သိချင်တဲ့ အခါ

In [25]:

```
whatis ssh
```

```
ssh (1)               - OpenSSH SSH client (remote login program)
```

ကြည့်တဲ့ command အပေါ်မူတည်ပြီး section တစ်ခုထက်မက ပိုရှိရင်၊ ရှိတဲ့ section အားလုံးကို ပြပေးပါလိမ့်မယ်။
ဥပမာ passwd ကို whatis နဲ့ကြည့်မယ်ဆိုရင်

In [26]:

```
whatis passwd
```

```
passwd (1)           - change user password
passwd (1ssl)        - compute password hashes
passwd (5)           - the password file
```

20. info (read Info documents)

info က man လိုမျိုး help (သို့) manual စာမျက်နှာကိုပြပေးတဲ့ နောက်ထပ် command တစ်ခုပါ။

ဘာကြောင့် man နဲ့ info ဆိုပြီးနှစ်မျိုးရှိနေရတာလဲ၊ man နဲ့ info က ဘာကွာသလဲဆိုတာကို အတိုရှင်းပြပါမယ်။ man က Unix လက်ထက်ကတည်းက သုံးခဲ့တဲ့ traditional ပုံစံပါ။ လက်ရှိ ဒီနေ့ခေတ်အထိ command line မှာ help ကြည့်ဖို့အတွက်က အရမ်းကို အသုံးဝင်ပါသေးတယ်။ သို့သော် အထက်မှာ သုံးပြခဲ့တဲ့ အတိုင်း man မှာက section တွေခွဲပြီး ဖိုင်တွေကိုလည်း သပ်သပ်စီခွဲသိမ်းထားပါတယ်။ ဥပမာ passwd ရဲ့ man page section 1 နဲ့ ဆိုင်တဲ့ဖိုင်ကို /usr/share/man/man1/passwd.1.gz မှာ section 5 နဲ့ ဆိုင်တဲ့ဖိုင်ကို /usr/share/man/man5/passwd.5.gz မှာ သိမ်းထားပါတယ်။ အဲဒါကြောင့် အားလုံးကို ပေါင်းပြီး ပရင့်ထုတ်တာမျိုးလုပ်ချင်တယ် ဆိုရင် သိပ်ပြီးတော့ အဆင်မပြေခဲ့ပါ။

၁၉၉၀ လောက်မှာ GNU (ဂနူ လို့အသံထွက်တယ်) ပရောဂျက်အဖွဲ့က man တွေကို အစားထိုးဖို့ info ဆိုတာကို စတင် လုပ်ဆောင်ခဲ့ရာက info help documents တွေပေါ် ပေါက်လာတာပါ။ info က markup language ကို သုံးထားပါတယ်။ အဲဒါကြောင့် hyperlink တွေပါရှိပြီး၊ အကြောင်းအရာ တစ်ခုကနေ နောက်အကြောင်းအရာ တစ်ခုစီကို ခုန်ကူးပြီး ကြည့်လို့ရပါတယ်။

အမှန်တကယ်က Linux မှာ help ကြည့်ဖို့အတွက်က man နဲ့ info နှစ်မျိုးထဲမဟုတ်ပါဘူး။ တခြား ပရောဂျက်ကွဲတွေဖြစ်တဲ့ GNOME, KDE တို့မှာလည်း HTML ကို အခြေခံထားတဲ့ help documentation တွေရှိကြပါသေးတယ်။ အဲဒါကြောင့် ကိုယ်သုံးတဲ့ Linux system ပေါ် မူတည်ပြီး help (သို့) manual တွေရဲ့ပုံစံက ကွဲပြားနိုင်ပါတယ်။

info ကိုသုံးပြီး command တချို့ရဲ့ help screen ကြည့်ပုံကို နမူနာအနေနဲ့ သုံးပြပါမယ်။
ဥပမာ vi text editor ကို info နဲ့ ကြည့်မယ်ဆိုရင်

info vi

vi ရဲ့ options တွေကို သိချင်တယ်ဆိုရင်

info --show-options vi

info ကို ဘာ option မှ မပေးပဲ အောက်ပါအတိုင်းရိုက်ရင်၊ info ကို သုံးထားတဲ့ application အားလုံးကို မာတိကာပုံစံနဲ့ တွေ့ရပါလိမ့်မယ်။

In [ ]:

```
info
```

ဘယ်လို ပုံစံနဲ့ မြင်ရသလဲဆိုတာကို idea ရအောင် screen ရဲ့တစိတ်တပိုင်းကိုပဲ အောက်ပါအတိုင်း ဖော်ပြပါမယ်။

*base32: (coreutils)base32 invocation. Base32 encode/decode data.
*base64: (coreutils)base64 invocation. Base64 encode/decode data.
*basename: (coreutils)basename invocation. Strip directory and suffix.
*bibtex: (web2c)bibtex invocation. Maintaining bibliographies.
*cat: (coreutils)cat invocation. Concatenate and write files.
*chcon: (coreutils)chcon invocation. Change SELinux CTX of files.
*chgrp: (coreutils)chgrp invocation. Change file groups.
*chmod: (coreutils)chmod invocation. Change access permissions.
*chown: (coreutils)chown invocation. Change file owners and groups.
*chroot: (coreutils)chroot invocation. Specify the root directory.
*cksum: (coreutils)cksum invocation. Print POSIX CRC checksum.

*cmp: (diffutils)Invoking cmp. Compare 2 files byte by byte.
*comm: (coreutils)comm invocation. Compare sorted files by line.
*cp: (coreutils)cp invocation. Copy files.
*csplit: (coreutils)csplit invocation. Split by context.
*cut: (coreutils)cut invocation. Print selected parts of lines.
*date: (coreutils)date invocation. Print/set system date and time.
*dd: (coreutils)dd invocation. Copy and convert a file.
*df: (coreutils)df invocation. Report file system disk usage.
*diff: (diffutils)Invoking diff. Compare 2 files line by line.
*diff3: (diffutils)Invoking diff3. Compare 3 files line by line.
*dir: (coreutils)dir invocation. List directories briefly.
*dircolors: (coreutils)dircolors invocation. Color setup for ls.
*dirname: (coreutils)dirname invocation. Strip last file name component.
*du: (coreutils)du invocation. Report on disk usage.
*dvicopy: (web2c)dvicopy invocation. Virtual font expansion
*dvitomp: (web2c)dvitomp invocation. DVI to MPX (MetaPost pictures).
*dvitype: (web2c)dvitype invocation. DVI to human-readable text.
*echo: (coreutils)echo invocation. Print a line of text.
*env: (coreutils)env invocation. Modify the environment.
*expand: (coreutils)expand invocation. Convert tabs to spaces.
*expr: (coreutils)expr invocation. Evaluate expressions.
*factor: (coreutils)factor invocation. Print prime factors

21. Escape Sequences

Escape sequence (backslash escape character လို့လည်းခေါ်) က ပရိုဂရမ်မင်းဘာသာစကားတွေဖြစ်တဲ့ C, Java, Perl တို့မှာ သုံးသလိုပါပဲ။ Escaping လုပ်ချင်တဲ့ စာလုံးကို "\" (backslash) ခံပြီးရိုက်ပါတယ်။ bash က "\" နောက်က စာလုံးကို interpret လုပ်တဲ့အခါမှာ သတ်မှတ်ထားတဲ့ escape sequence တွေအတိုင်း အလုပ်လုပ်ပေးပါတယ်။

Bash shell မှာသုံးတဲ့ escape sequences တွေက အများကြီးရှိပါတယ်။ အဲဒီအထဲက အသုံးများတဲ့ escape sequence တချို့နဲ့ သူတို့ရဲ့အလုပ်လုပ်ပုံကတော့ အောက်ပါဇယားအတိုင်း ဖြစ်ပါတယ်။

Escape sequence	အလုပ်လုပ်ပုံ
\a	သတိပေးချင်တဲ့ အခါ၊ စပီကာကနေ bell အသံပေးချင်တဲ့အခါ သုံးတယ်။ ASCII bell character လို့လည်းခေါ်တယ်။
\b	Backspace ကီးကို ရိုက်ချင်တဲ့အခါ သုံးတယ်။
\c	သူနောက်က စာလုံးတွေကို ရိုက်မပေးတော့ဘူး။ နောက်ပြီး စာကြောင်းအသစ်တစ်ကြောင်းအနေနဲ့လည်း ခွဲမပေးပါဘူး
\f	form feed (စာမျက်နှာ အသစ်ခွဲ) အလုပ်ကို လုပ်ပေးတယ်။
\n	လက်ရှိ ရောက်ရှိနေတဲ့နေရာကနေ စာကြောင်းအသစ်ရဲ့ဘယ်ဘက် အကျဆုံးနေရာကို ရွှေ့ပေးလိမ့်မယ်။
\r	Carriage return သင်္ကေတပါ။ လက်ရှိစာကြောင်းရဲ့ဘယ်ဘက်ထိပ်ဆုံး နေရာကို ရွှေ့ချင်တဲ့အခါ အသုံးပြုတယ်။
\t	Horizontal Tab ကီးကို ရိုက်ဖို့အတွက် သုံးတယ်။
\v	Vertical Tab ကီးကို ရိုက်ဖို့အတွက် သုံးတယ်။
\[	စကရင်မှာ ရိုက်ပြဖို့မဟုတ်ပဲ၊ တစ်ခြား အလုပ်လုပ်ဖို့ သတ်မှတ်ထားသောစာလုံး (non-printing character)တွေရဲ့ကွင်းစ
\]	စကရင်မှာ ရိုက်ပြဖို့မဟုတ်ပဲ၊ တစ်ခြား အလုပ်လုပ်ဖို့ သတ်မှတ်ထားသောစာလုံး (non-printing character) တွေရဲ့ကွင်းပိတ်
\'	single quote ကို ရိုက်ဖို့အတွက်သုံးတယ်။
\"	double quote ကို ရိုက်ဖို့အတွက်သုံးတယ်။

Escape sequence	အလုပ်လုပ်ပုံ
\nnn	eight-bit စာလုံးတွေကို octal value နဲ့ ရိုက်ဖို့အတွက်သုံးတယ်။
\xHH	eight-bit စာလုံးတွေကို hexadecimal value နဲ့ ရိုက်ဖို့အတွက်သုံးတယ်။
\uHHHH	Unicode (ISO/IEC 10646) စာလုံးတွေကို hexadecimal value နဲ့ ရိုက်ဖို့သုံးပါတယ်။ (hexadecimal ဂဏန်း 4 လုံးအထိ)
\UHHHHHHHH	Unicode (ISO/IEC 10646) စာလုံးတွေကို hexadecimal value နဲ့ ရိုက်ဖို့သုံးပါတယ်။ (hexadecimal ဂဏန်း ၈ လုံးအထိ)
\cx	control ကီးနဲ့ တခြားကီးတစ်ခုခုကို တွဲရိုက်ဖို့ (control-x) အတွက် သုံးပါတယ်။

bash command တွေနဲ့ escape sequence သုံးပုံတချို့ကို ဥပမာပေးပါမယ်။

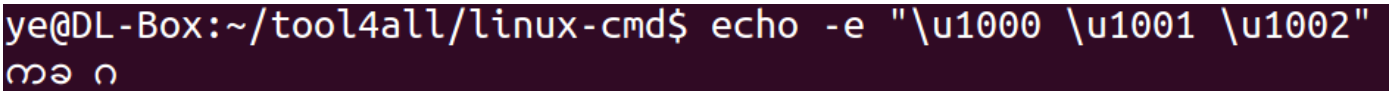
က၊ ခ၊ ဂ ရဲ့ Unicode code နံပါတ်တွေက u1000, u1001 နဲ့ u1002 အသီးသီး ဖြစ်ကြပါတယ်။
echo command နဲ့ က ခ ဂ ကို ရိုက်ပြချင်တဲ့အခါ၊ အောက်ပါအတိုင်း escape sequence \u ကိုသုံးပြီး ရိုက်ခိုင်းလို့ ရပါတယ်။

In [1]:

```
echo -e "\u1000 \u1001 \u1002"
```

က ခ ဂ

jupyter notebook ရဲ့ bash kernel က တစ်ခါတလေမှာ မြန်မာစာလုံးတွေကို သေသေချာချာမပြပေးနိုင်လို့ အောက်ပါ screen capture လုပ်ထားတဲ့ ပုံကို ပါထည့်ပေးထားပါတယ်။



ဒီနေရာ မှာ သိထားရမှာက echo command ရဲ့ -e option ကိုပါတွဲသုံးရတယ်ဆိုရတာကိုပါ။
-e option မပါရင် စာလုံးတွေကို ဒီအတိုင်းပဲ ပြန်ရိုက်ပေး (literally print) လုပ်ပေးပါလိမ့်မယ်။

In [2]:

```
echo "\u1000 \u1001 \u1002"
```

```
\u1000 \u1001 \u1002
```

ကွန်ပြူတာရဲ့ စပီကာကနေ သတိပေးတဲ့အနေနဲ့ bell အသံထုတ်ပေးချင်တဲ့အခါ၊ အထက်ပါ ဇယားမှာ ရှင်းပြခဲ့တဲ့ အတိုင်း "\a" escape sequence ကိုသုံးပြီး လုပ်ခိုင်းလို့ရပါတယ်။

In [3]:

```
echo -e "\a \a \a"
```

□ □ □

ဒီနေရာမှာ တယ်လီဖုန်းပုံ (သို့) လေးထောင့်ကွက်အနေနဲ့ ရိုက်ပြတာက၊ jupyter notebook ကိုသုံးထားလို့၊ output ကိုမြင်သာအောင် ပုံအနေနဲ့ jupyter notebook ကထုတ်ပေးတာ သာဖြစ်ပါတယ်။ တကယ်တမ်း linux ရဲ့ bash shell မှာ run ရင် မြင်ရမှာ မဟုတ်ပါဘူး။

အသုံးများတဲ့ escape sequence တွေက သိထားသင့်ပါတယ်။ တကယ်တမ်းက escape sequence တွေကို စကရင်မှာ ရိုက်ထုတ်တဲ့ ကိစ္စတင် မကပဲ၊ ဖိုင်ထဲကစာသားတွေ၊ user ကရိုက်ထည့်လိုက်လို့ ဝင်လာတဲ့ စာသားတွေမှာ ရှာဖွေတဲ့ ကိစ္စ၊ ဝင်ပြင်တဲ့ ကိစ္စတွေမှာလည်း သုံးနိုင်ပါတယ်။

အောက်ပါ ဥပမာက နဂို fileA မှာရှိတဲ့ space နေရာတွေမှာ \t ကိုသုံးပြီး အစားထိုးတာပါ။
fileA မှာဘာစာကြောင်းတွေရှိသလဲ ဆိုတာကို cat နဲ့ ရိုက်ထုတ်ကြည့်မယ်။

In [4]:

```
cat fileA
```

```
Mingalar bar!
I am fileA.
I was born in April.
My blood type is O.
```

space (\s) ကို tab (\t) နဲ့ အစားထိုးတာကို sed command နဲ့ လုပ်ကြည့်မယ်။

In [6]:

```
cat ./fileA | sed 's/\s/\t/g'
```

```
Mingalar	bar!
I	am	fileA.
I	was	born	in	April.
My	blood	type	is	O.
```

ဒီနေရာမှာ s/\s/\t/g က Regular Expression ပါ။
ပထမဆုံး "s" က စာလုံးတွေ၊ စာကြောင်းတွေကို အစားထိုးချင်တဲ့အခါ သုံးတဲ့ "substitute လုပ်ပေးပါ" ဆိုတဲ့ အဓိပ္ပါယ်ပါ။
"\s" က space ကို ဆိုလိုပါတယ်။ "\t" က tab ကိုဆိုလိုပါတယ်။ နောက်ဆုံးက "g" ကတော့ global အနေနဲ့ အလုပ်လုပ်ပေးပါလို့ ဆိုလိုပါတယ်။ "g" ကို နောက်ဆုံးမှာ မထည့်ရင် စာကြောင်းတစ်ကြောင်းမှာ ပထမဆုံးတွေ့တဲ့ space ကိုပဲ tab နဲ့ အစားထိုးပေးပါလိမ့်မယ်။

အောက်ပါ ဥပမာက "g" မပါပဲ run တဲ့ အခါတွေ့ရမဲ့ output ပါ။

In [7]:

```
cat ./fileA | sed 's/\s/\t/'
```

```
Mingalar	bar!
I	am fileA.
I	was born in April.
My	blood type is O.
```

sed command ရဲ့သုံးပုံသုံးနည်းနဲ့ Regular Expression ရဲ့သုံးပုံသုံးနည်းကို နောက်ပိုင်းမှာ ထပ်ရှင်းပြပါမယ်။

22. echo (display a line of text)

echo က စာကြောင်းကို မော်နီတာ စကရင်မှာ ရိုက်ပေးတဲ့ command ပါ။
Operating System (OS) တိုင်းလိုလိုမှာ အခြေခံကျတဲ့ command တစ်ခုအနေနဲ့ ပါဝင်ပြီး၊ ပရိုဂရမ်မင်း ဘာသာစကားတွေမှာပါတဲ့ print, printf, cout, writeln, puts တို့နဲ့ တူပါတယ်။

echo command ရဲ့ syntax ကတော့ အောက်ပါအတိုင်းဖြစ်ပါတယ်။
echo [option(s)] [string(s)]

အသုံးပြုပုံ ဥပမာတချို့ကတော့ အောက်ပါအတိုင်း ဖြစ်ပါတယ်။

In [8]:

```
echo Hello Aliean!
```

```
Hello Aliean!
```

In [9]:

```
X=88; Y=91;
echo X is $X and Y is $Y.
```

```
X is 88 and Y is 91.
```

In [10]:

```
echo -e "Who\nare\nyou\n?"
```

```
Who
are
you
?
```

Escape sequences တွေကိုပါ စာရိုက်ခိုင်းတဲ့အခါမှာသုံးချင်ရင် အထက်မှာ ပြထားတဲ့ အတိုင်း -e option ကို သုံးရပါတယ်။ နောက်ထပ် ဥပမာ တစ်ခုထပ်ပေးရင်၊ \t က TAB ကီးကို ကိုယ်စားပြုပါတယ်။

In [11]:

```
echo -e "Who\tare\tyou\t?"
```

```
Who	are	you	?
```

In [12]:

```
echo -e "I am a \\ (backslash).\nNice to see you!"
```

```
I am a \ (backslash).
Nice to see you!
```

တကယ်လို့ ကျွန်တော်တို့ echo နဲ့ရိုက်ခိုင်းမဲ့ စာကြောင်းက ရှည်နေလို့ မော်နီတာစကရင်မှာ ကြည့်ရတာအဆင်ပြေအောင် နောက်လိုင်းတကြောင်းဆင်းပြီးတော့ ရိုက်ပြပေးစေချင်ရင် "\" (backslash) ရိုက်ပြီး enter ကီးနှိပ်ရင် နောက်တကြောင်းကို ဆင်းပေးပါလိမ့်မယ်။

ဒီနေရာမှာ သတိထားရမှာက "\" နောက်မှာ space ကီး စတဲ့ တခြားစာလုံးတွေကို မရိုက်မိစေပဲ၊ "\" ကိုရိုက်ပြီးတာနဲ့ enter ကီးကို ခေါက်ပေးဖို့လိုအပ်ပါတယ်။

In [19]:

```
echo "This is, this is, this is, this is, this is, this is \
on two lines"
```

```
This is, this is, this is, this is, this is, this is on two lines
```

shell (သို့) command prompt မှာ "`" (backtick) ကို သုံးပြီး command တစ်ခုကို run ခိုင်းပြီးတော့ ရလာတဲ့ ရလဒ် (output) ကို ယူသုံးလို့ရပါတယ်။ ကိုယ်က run စေချင်တဲ့ command နဲ့ သူနဲ့ ဆိုင်တဲ့ option, parameter စတာတွေကို backtick နှစ်ခုကြားမှာ ထည့်ပြီး run ခိုင်းတဲ့ ပုံစံပါ။

shell က main command ကို execute မလုပ်ခင်မှာ backtick နှစ်ခုအတွင်းမှာရှိတဲ့ command ကို အရင်ဆုံး evaluate လုပ်တယ်။ ပြီးတော့မှ ရလာတဲ့ ရလဒ်ကို main command ရဲ့အစိတ်အပိုင်းတခုအဖြစ် ရိုက်ထုတ်ပြီး၊ main command ကို run ပါတယ်။

အောက်ပါ ဥပမာက backtick နှစ်ခုကြားမှာ ရှိတဲ့ "date" command ကို အရင် run ပြီးမှ ရလာတဲ့ ရက်စွဲကို echo က ရိုက်ထုတ်ပြတာကို လုပ်ပြတာပါ။

In [2]:

```
echo the date is `date`
```

```
the date is 2017年 8月 26日 土曜日 10:52:09 JST
```

backtick ကို မသုံးပဲ $(command) ဆိုတဲ့ ပုံစံလည်းရှိပါတယ်။
အောက်ပါ command ကို run ကြည့်ရင် backtick ကို သုံးခဲ့သလိုပဲ အတူတူ အလုပ်လုပ်ပေးတာကို တွေ့ရပါလိမ့်မယ်။

In [1]:

```
echo the date is $(date)
```

```
the date is 2017年 8月 26日 土曜日 10:52:05 JST
```

echo command ကို video text terminal တွေရဲ့ format, color, cursor movement စတာတွေအတွက် သတ်မှတ်ထားတဲ့ ANSI (American National Standard Institute) escape code တွေနဲ့ စာကြောင်းထဲမှာ တွဲသုံးပြီးတော့ ရိုက်ပေးမဲ့ စာလုံးတွေရဲ့အရောင်၊ နောက်ခံအရောင်တွေကိုလည်း ပြောင်းလို့ရပါတယ်။

In [38]:

```
echo -e "\u001b[32;1m Now GREEN text! \u001b[0m Right? "
```

```
 Now GREEN text!  Right?
```

In [16]:

```
echo -e "\u001b[44;1m Nyein \u001b[41;1m Aye \u001b[47;1m Thu \u001b[42;1m is \u001
```

```
 Nyein  Aye  Thu  is  a  girl.
```

အထက်မှာ သုံးပြခဲ့တဲ့ echo command တွေမှာ ပါတဲ့ \u001b ဆိုတာက Escape အတွက် သတ်မှတ်ထားတဲ့ Unicode Code နံပါတ် ဖြစ်ပါတယ်။
Unicode နံပါတ်တွေကို အမျိုးမျိုး သုံးပြီး echo နဲ့ terminal မှာ ရိုက်ပြလို့ရပါတယ်။

မြန်မာဗျည်း ကကြီးကို echo command နဲ့ unicode escape sequence \u1000 ကို သုံးပြီး terminal မှာ ရိုက်ကြည့်ရအောင်။

In [17]:

```
echo -e "Unicode number U+1000 is \"\U1000\" (a Myanmar consonant Ka Gyi)."
```

```
Unicode number U+1000 is "က" (a Myanmar consonant Ka Gyi).
```

Jupyter notebook ရဲ့ bash command output အနေနဲ့ မပြပေးနိုင်လို့ ပုံအနေနဲ့ ထည့်ပေးထားတာပါ။ terminal မှာ run ကြည့်ရင် အောက်ပါအတိုင်း မြင်ရပါလိမ့်မယ်။

Bayes' theorem ကို linux terminal မှာ အောက်ပါအတိုင်း echo command နဲ့ Unicode နံပါတ်တချို့ကို သုံးပြီး ရိုက်ကြည့်ပါ။
echo -e "\u2119(A\u007CB)=(\u2119(B\u007CA)\u2119(A))\u2215\u2119(B)"
ဒီနေရာမှာ \u2119 က Probability mesure "P" ရဲ့ unicode ပါ။
\u007C က conditional probability "|" သင်္ကေတရဲ့ unicode နံပါတ်ပါ။
အောက်ပါအတိုင်းမြင်ရပါလိမ့်မယ်။

ဒီနေရာကနေစပြီး linux command တွေကို သုံးတဲ့ နေရာမှာတင်မက၊ programmering လုပ်တဲ့အခါ၊ networking အလုပ်တွေ လုပ်တဲ့အခါ၊ နေရာတိုင်းမှာ ပုံစံအမျိုးမျိုးနဲ့သုံးကြတဲ့ wild card, brace expansion, redirection, regular expression (RE) တွေကို concept ရအောင်၊ အကြမ်းမျဉ်းရင်းပြပါမယ်။

23. Wild Card (*?) and Brace Expansion [...], {...}

Linux/Unix မှာ ဖိုင်တွေအမြောက်အမြားနဲ့ အလုပ်လုပ်နိုင်ဖို့အတွက် wild card နဲ့ brace expansion ဆိုတာကို သုံးပါတယ်။

Wild Card

- Question Mark Wild card ("?") က စာလုံး တစ်လုံးကို ကိုယ်စားပြုပါတယ်။ ဖိုင်စနစ်က အသိအမှတ်ပြုထားတဲ့ ကြိုက်တဲ့ စာလုံးဖြစ်လို့ရပါတယ်။

- Star wild cad ("*") က စာလုံး အရေအတွက် သုည ကနေ အရေအတွက် အကန့်အသတ်မရှိတဲ့အထိ ကိုယ်စားပြုပါတယ်။ ကြိုက်တဲ့ စာလုံးဖြစ်လို့ ရပါတယ်။
  တနည်းအားဖြင့် "*" က တခြား wild card တွေအားလုံးကိုပါ ကိုယ်စားပြုတယ်လို့ ပြောလို့ရပါတယ်။

Brace Expansion

- Brace expansion [...] က လေးထောင့်ကွင်း အဖွင့် "[" သင်္ကေတ နဲ့ အပိတ်ဖြစ်တဲ့ "]" သင်္ကေတ နှစ်ခုအကြားမှာ ရာဖွေချင်တဲ့ စာလုံးအတွဲလိုက်၊ ဘယ်စာလုံးကနေ ဘယ်စာလုံးအထိ ဆိုတာမျိုး ကို သတ်မှတ်ပြီး ရာခိုင်းလို့ရပါတယ်။ ဘယ်စာလုံးကနေ ဘယ်စာလုံးရဲ့ အတွင်းမှာ ဆိုတာကိုတော့ "-" dash စာလုံးနဲ့ ပိုင်းခြားပေးရပါတယ်။ ဥပမာ [a-z] ဆိုတာက a ကနေ z အတွင်းမှာရှိတဲ့ အင်္ဂလိပ်စာလုံး တစ်လုံးလို့ ဆိုလိုတာပါ။ စာလုံးတွဲတွေကို တခုထက်မက လေးထောင့်ကွင်းအတွင်းမှာ ထည့်ပြီး ညွှန်းလို့ရပါတယ်။ ဥပမာ [A-Da-d0-5] ဆိုတာက [ABCDabcd012345] နဲ့ သွားညီပါလိမ့်မယ်။ ဒါအပြင် လေးထောင့်ကွင်း အဖွင့် "[" နဲ့ အပိတ်ဖြစ်တဲ့ "]" အကြားမှာ ဒီ စာလုံး (သို့) ဒီစာလုံး ကို ရာဖွေပေးပါဆိုတဲ့ ပုံစံမျိုးလဲ သုံးနိုင်ပါတယ်။ ဥပမာ [137] ဆိုတဲ့ brace expansion က အင်္ဂလိပ်နံပါတ် 1 (သို့) 3 (သို့) 7 တစ်ခုခု လို့ ညွှန်းတာဖြစ်ပါတယ်။

- Brace expansion {...} က တွန့်ကွင်း အဖွင့် "{" သင်္ကေတနဲ့ အပိတ်သင်္ကေတ "}" နှစ်ခုအကြားမှာ ပေးလိုက်တဲ့ သတ်မှတ်ချက်ကို အခြေခံပြီး၊ စာလုံးတွေကို command line (သို့) shell script ရဲ့အတွင်းမှာ ထုတ်လုပ်ပေးနိုင်၊ အင်္ဂလိပ်လိုဆိုရင် generate လုပ်ပေးနိုင်ပါတယ်။ တကယ်တမ်း အသုံးပြုတဲ့အခါမှာ brace expansion ရဲ့ ရှေ့မှာရှိတဲ့ စာလုံး (prefix) နဲ့ brace expansion ရဲ့ နောက်မှာရှိတဲ့ စာလုံး (suffix) တွေနဲ့ ပေါင်းစပ်ပြီး ရိုက်ထုတ်ပေးလို့ အရမ်းအသုံးဝင်ပါတယ်။ ဘယ်စာလုံးကနေ ဘယ်စာလုံးအထိ ဆိုတာကို ".." နဲ့ ရေးပါတယ်။ ဥပမာ {0..5} လို့ရေးရင် 0 1 2 3 4 5 ဆိုပြီး generate လုပ်ပေးပါလိမ့်မယ်။ တကယ်လို့ A{0..3}B လို့ရေးရင် A0B, A1B, A2B နဲ့ A3B ကို generate လုပ်ပေးပါလိမ့်မယ်။ ဘယ်စာလုံးကနေ ဘယ်စာလုံးအထိ ဆိုတာမျိုး မဟုတ်ပဲ၊ စာလုံး တစ်လုံးချင်းစီ သပ်သပ် ဆိုရင်တော့ ကော်မာ "," နဲ့ ခြားပြီး ပေးပါတယ်။ ဥပမာ {dog, flower, wine} လို့ရေးပါတယ်။

  တစ်ခု သိထားရမှာက လေးထောင့်ကွင်း၊ တွန့်ကွင်းတွေရဲ့အတွင်းမှာ "?" နဲ့ "*" wild card တွေကို ထည့်သုံးလို့ မရပါ။ bash shell က interpret လုပ်ပေးနိုင်မှာ မဟုတ်ပါဘူး။

အထက်ပါရှင်းပြခဲ့တဲ့ wild card တွေကို အောက်ပါအတိုင်း terminal မှာ စမ်းရိုက်ကြည့်ပါ။

final:
Wild card စာလုံးတွေက shell တွေအပေါ်မှာ မူတည်ပြီး သတ်မှတ်ချက်တွေက မတူညီကြပါဘူး။ ဥပမာ bash shell မှာ သုံးတဲ့ wild card စာလုံးတွေအားလုံးက zsh shell မှာ အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။

Wild card နဲ့ ပတ်သက်ပြီး အထက်ပါရှင်းပြခဲ့တာကို သဘောပေါက်တယ်ရင် တကယ့်လက်တွေမှာလည်း အသုံးချနိုင်ပါလိမ့်မယ်။ သုံးလည်း သုံးစေချင်ပါတယ်။
နောက်ဆုံးအနေနဲ့ bash shell မှာသုံးတဲ့ wild card နဲ့ ပတ်သက်ပြီး အခြေခံကို အင်္ဂလိပ်လို အတိုရှင်းပြပါဆိုရင် အောက်ပါအတိုင်း ဖြစ်ပါလိမ့်မယ်။

* <code>*</code> (matches anything)
* ? (matches any single character)
* [xyz] (matches any of x, y, or z)
* [a-d] (matches any character in the range a through d)

လက်ရှိ ရောက်နေတဲ့ path မှာရှိတဲ့ ဖိုင်တွေကို list လုပ်ကြည့်ရအောင်။

In [18]:

```
ls
```

```
all-vi-info.txt  fileA         item-list                 otest
data1            fileB         linux-commands.ipynb      otest.tag
data2            fmt.out       linux-commands.ipynb.bak  otest.word
data3            folderA       ls.out                    screen
data4            folder-athit  my-news                   sorted-names
data5            folderB       newfile                   wildcard
echo             folder-new    news
```

Wild card "?" ကို စသုံးကြည့်မယ်။

In [19]:

```
ls file?
```

```
fileA  fileB
```

နောက်ပုံစံတစ်မျိုး သုံးကြည့်မယ်။

In [20]:

```
ls ????
```

```
echo  news
```

? လေးခုပေးခဲ့တာဖြစ်လို့ စာလုံးလေးလုံးရှိတဲ့ ဖိုင်နာမည် news ကို ရိုက်ထုတ်ပြပါတယ်။

Wild card "*" ကို သုံးကြည့်ရအောင်။

In [21]:

```
ls file*
```

```
fileA  fileB
```

In [22]:

```
ls f*
```

```
fileA  fileB  fmt.out

folderA:
127-135.pdf

folder-athit:

folderB:
127-135.pdf  fileA  otest

folder-new:
folder-new
```

f"*" ဖြစ်တဲ့ အတွက်ကြောင့်၊ fနဲ့စတဲ့ဖိုင်၊ ဖိုလ်ဒါတွေကို ရိုက်ထုတ်ပေးပါလိမ့်မယ်။ ဖိုလ်ဒါဆိုရင်တော့ အဲဒီဖိုလ်ဒါအတွင်းမှာရှိတဲ့ ဖိုင်တွေကိုပါ ရိုက်ထုတ်ပြပါလိမ့်မယ်။

In [23]:

```
cd wildcard
```

In [24]:

```
ls *m*
```

```
acknowledgements.txt  m  maik-khe.food.txt  matrix  mba.note.txt  supe
rman
```

အထက်ပါ အမိန့် "*m*" က လက်ရှိဖိုလ်ဒါအောက်ရှေ့ကစချင်တာနဲ့ စ၊ "m" စာလုံးပါပြီး နောက်မှာလည်း ဆုံးချင်တာနဲ့ ဆုံးတဲ့ ဖိုင်ကို ပြပေးပါလို့ ဆိုလိုပါတယ်။ ဒီနေရာမှာ သိထားရမှာက wild card * က ရှေ့မှာ ရှင်းပြခဲ့သလို စာလုံးအရေအတွက် သုညကနေ အရေအတွက် အကန့်အသတ်မရှိတဲ့အထိ ကိုယ်စားပြု တာမို့၊ m ရဲ့ရှေ့မှာ ဘာစာလုံးမှမရှိတာ၊ m ရဲ့နောက်မှာ ဘာစာလုံးမှမရှိတာလဲ ပါဝင်ပါတယ်။ အဲဒါကြောင့် ဖိုင်နာမည် m လိုဖိုင်မျိုးကိုပါ ရိုက်ထုတ်ပြတာပါ။

Wild card "*" ကို echo command နဲ့ သုံးကြည့်ရအောင်။
echo * command နဲ့ လက်ရှိရှိနေတဲ့ ဖိုလ်ဒါအောက်က ဖိုင်တွေ၊ ဖိုလ်ဒါတွေအားလုံးကို ရိုက်ခိုင်းလို့ရပါတယ်။ ls command လိုပဲ အလုပ်လုပ်ပေးပါတယ်။

In [25]:

```
echo *
```

```
abstract.txt acknowledgements.txt city conclusion.txt graph1 Graph1 gr
aph2 Graph2 graph3 Graph3 graph4 Graph4 graph5 hello.txt m maik-khe.fo
od.txt matrix mba.note.txt superman
```

echo g* နဲ့ gနဲ့စတဲ့ ဖိုင်၊ ဖိုလ်ဒါတွေကို ရာကြည့်လို့ရပါတယ်။ လက်ရှိ ဖိုလ်ဒါpath မှာက gနဲ့စတဲ့ ဖိုလ်ဒါတွေပဲ ရှိလို့၊ ဖိုလ်ဒါတွေကိုပဲ ရိုက်ထုတ်ပြပါလိမ့်မယ်။

In [28]:

```
echo g*
```

```
graph1 graph2 graph3 graph4 graph5
```

echo ရဲ့အဓိကအလုပ်က input လုပ်တဲ့ စာကြောင်းတွေ ကို ရိုက်ထုတ်ပြတာမို့လို့ ls command လို ဖိုင်တွေ၊ ဖိုလ်ဒါတွေရဲ့အသေးစိတ် အချက်အလက်တွေကို ပေးမှာမဟုတ်ပါဘူး။

Wild card ? နဲ့ * နှစ်ခုကို ပေါင်းပြီး သုံးတဲ့ ဥပမာတချို့ကတော့ အောက်ပါအတိုင်းပါ။

In [32]:

```
ls ../o*.???
```

```
../otest.tag
```

အထက်ပါ "ls o*.???" က အင်္ဂလိပ်စာလုံး o (small o) နဲ့ စပြီး၊ ဖိုင်တွေရဲ့အမျိုးအစားတွေကို ခွဲတဲ့နေရာမှာ သုံးလေ့ရှိကြတဲ့ "." (dot) လဲ ပါပြီး၊ အဲဒီ . (dot) ရဲ့နောက်မှာတော့ စာလုံးသုံးလုံးနဲ့ ဆုံးတဲ့ ဖိုင်၊ ဖိုလ်ဒါတွေ ကို လက်ရှိဖိုလ်ဒါအောက်မှာ ရှာပေးပါလို့ ခိုင်းတဲ့ အမိန့်ပါ။

လက်ရှိဖိုလ်ဒါအောက်မှာရှိတဲ့ ဖိုင်တွေ၊ ဖိုလ်ဒါတွေရဲ့နာမည်တွေအားလုံးအထဲမှာမှ ပေးလိုက် တဲ့ သတ်မှတ်ချက်ဖြစ်တဲ့ "o*.???"နဲ့ ကိုက်ညီတဲ့ "otest.tag" ဖိုင်ကို ရိုက်ထုတ်ပြတာပါ။

In [37]:

```
ls -lh /home/*/.bash_history
```

```
-rw------- 1 guest guest  21  8月 11 20:20 /home/guest/.bash_history
-rw------- 1 lar   lar   28K  8月 11 19:58 /home/lar/.bash_history
```

အထက်ပါ /home/*/ က home ဖိုလ်ဒါအောက်မှာ ရှိတဲ့ username နဲ့သိမ်းထားတဲ့ ဖိုလ်ဒါအားလုံးကို ညွှန်းတာပါ။ /.bash_history က ပုံမှန်အားဖြင့်မမြင်ရတဲ့ ဖွက်ထားတဲ့ဖိုင် (hidden file) ဖြစ်တဲ့ bash_history (ရိုက်ခဲ့သမျှ bash command တွေကိုသိမ်းထားတဲ့ ဖိုင်) ကို ပြောတာပါ။

Brace expansion တွေကို စမ်းသုံးကြည့်ကြရအောင်။

In [39]:

```
ls graph[1-5]
```

```
graph1  graph2  graph3  graph4  graph5
```

အထက်ပါ command က ဖိုင်နာမည် graph1 ကနေ graph5 အထိကို ls နဲ့ ပြခိုင်းကြည့်တာပါ။
လက်ရှိဖိုလ်ဒါအောက်မှာ ရှိရင် ပြပေးပါလိမ့်မယ်။

In [40]:

```
ls m[ab]*
```

```
maik-khe.food.txt  matrix  mba.note.txt
```

အထက်ပါ command မှာ ပါတဲ့ "m[ab]*" က "m" နဲ့ စပြီး သူ့ရဲ့နောက်က "a" စာလုံး (သို့) "b" စာလုံးက ကပ်လိုက်ပြီး၊ အဲဒီနောက်မှာတော့ ဘာပဲဖြစ်ဖြစ် ဆိုတဲ့ အဓိပ္ပါယ်ရပါတယ်။ အဲဒါကြောင့် လက်ရှိဖိုလ်ဒါအောက်မှာ၊ အဲဒီ brase expansion နဲ့ ကိုက်ညီတဲ့ maik-khe.food.txt ဖိုင်၊ matrix ဖိုင်နဲ့ mba.note.txt ဖိုင်တွေကို ရိုက်ပြပေးတာပါ။

In [41]:

```
echo {Kachin,Kayah,Kayin,Chin,Mon,Rakhine,Shan}\ State,
```

```
Kachin State, Kayah State, Kayin State, Chin State, Mon State, Rakhine
State, Shan State,
```

တစ်ခု သိထားရမှာက {} အတွင်းမှာ ရှိတဲ့ "," တွေရဲ့နောက်မှာ ပုံမှန်အင်္ဂလိပ်စာရိုက်သလိုမျိုး space ကီးကို ရိုက်ထည့်ရင် အလုပ်မလုပ်ပေးဘူးဆိုတာကိုပါ။
State စာလုံးရဲ့ ရှေ့မှာရှိတဲ့ "\ "က space ကိုရိုက်ပေးစေချင်လို့ escape လုပ်ထားတာပါ။

".." နဲ့ ဘယ်စာလုံးကနေ ဘယ်စာလုံးအထိ၊ ဘယ်ဂဏန်းကနေ ဘယ်ဂဏန်းအထိ ဆိုပြီး တိုးတဲ့အခါမှာ၊ ဘယ်နှစ်လုံးစီတိုးပေးပါဆိုတာကိုလည်း သတ်မှတ်ပေးလို့ရပါတယ်။ အောက်ပါ ဥပမာကို ကြည့်ကြရအောင်။

In [42]:

```
echo {-10..10..2}
```

```
-10 -8 -6 -4 -2 0 2 4 6 8 10
```

အထက်ပါ ဥပမာကို နားလည်မယ်လို့ ထင်ပါတယ်။
-10 ကနေ +10 အကြား၊ +2 လုပ်ပါ လို့ခိုင်းထားတာပါ။

In [43]:

```
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
```

အထက်ပါ brase expansion {z..a} က အင်္ဂလိပ်စာလုံး z ကနေ a အထိကို ပြောင်းပြန်အစီအစဉ်အတိုင်းရိုက်ခိုင်းတာပါ။

နောက်တစ်ချက် သိထားသင့်တာက brace expansion တွေက nested လုပ်လို့ရတယ်ဆိုတာကိုပါ။
Nested လုပ်တယ်ဆိုတာက brace expansion တစ်ခုရဲ့အတွင်းမှာ နောက်ထပ် တစ်ခြား brace expansion တစ်ခုကိုထည့်ရေးတာကိုပြောတာပါ။
အောက်ပါ ဥပမာကို ကြည့်ပါ။

In [44]:

```
echo {a{1..3},b{1..3},c{1..3}}
```

```
a1 a2 a3 b1 b2 b3 c1 c2 c3
```

အထက်ပါ ဥပမာမှာ အပြင်ဘက်အကျဆုံး brace expansion {} ရဲ့အတွင်းမှာ {1..3} ဆိုတဲ့ brace expansion ကို သုံးခါ ထည့်သုံးပြထားတာပါ။
output ကိုကြည့်ချင်းအားဖြင့် ဘယ်လိုအလုပ်လုပ်သွားတယ်ဆိုတာကို နားလည်ပါလိမ့်မယ်။
ဒီနေရာမှာလည်း ထပ်သတိပေးချင်တာက၊ "," ရဲ့နောက်မှာ space မရိုက်မိဖို့ပါ။

bash shell မှာ အလုပ်လုပ်တဲ့အခါ "ဒီစာလုံးတွေမဟုတ်တဲ့"၊ တစ်ခါတစ်လေမှာကျတော့ "ဒီစာလုံးကနေ ဒီစာလုံးကြား မဟုတ်တဲ့" ဆိုတဲ့ သတ်မှတ်ချက်မျိုးကို သတ်မှတ်ချင်တဲ့အခါ၊ brace expansion လေးထောင့်ကွင်းအဖွင့် သင်္ကေတ "[" ရိုက်ပြီးတာနဲ့ ! (exclamation mark) (သို့) ^ (caret) ကိုရိုက်ပြီး အဲဒီနောက်မှာ စာလုံးတွေကို ရိုက်ပြီး သုံးလေ့ရှိပါတယ်။

ဥပမာ rm [!0-9]* ဆိုရင် နံပါတ်နဲ့ မစတဲ့ ဖိုင်တွေအားလုံးကို ဖျက်ပေးပါလို့ ခိုင်းတာပါ။
(ဒီ command ကို သတိထားပြီးသုံးပါ။ လက်ရှိရောက်နေတဲ့ path အောက်မှာရှိတဲ့၊ နံပါတ်နဲ့ မစတဲ့ ဖိုင်တွေအားလုံးကို ဖျက်သွားပါလိမ့်မယ်။)

! ကို စမ်းသုံးပြပါမယ်။
လက်ရှိ ရောက်နေတာက /home/ye/tool4all/linux-cmd/wildcard/ ဖိုလ်ဒါအောက်မှာပါ။
အဲဒီ ဖိုလ်ဒါအောက်မှာ ဖိုင်တချို့ကို ကျွန်တော်ကြိုပြင်ထားပါတယ်။ အဲဒီဖိုင်တွေအားလုံးအထဲကနေ G (သို့) g နဲ့ စတဲ့ ဖိုင်တွေအားလုံးကို list လုပ်ကြည့်ပါမယ်။

In [45]:

```
ls [Gg]*
```

```
graph1  Graph1  graph2  Graph2  graph3  Graph3  graph4  Graph4  graph5
```

အထက်ပါ output အတိုင်း အင်္ဂလိပ်စာလုံး G အကြီး၊ g အသေးနဲ့ စတဲ့ဖိုင်တွေကို မြင်ရပါလိမ့်မယ်။
ဒီတစ်ခါတော့ ပြောင်းပြန် G (သို့) g နဲ့မစတဲ့ ဖိုင်တွေကိုပဲ ပြပေးပါဆိုပြီး [!Gg] wild card ကို သုံးပြီးလုပ်ခိုင်းပါမယ်။

In [46]:

```
ls [!Gg]*
```

```
abstract.txt          conclusion.txt  maik-khe.food.txt  superman
acknowledgements.txt  hello.txt       matrix
city                  m               mba.note.txt
```

! (exclamation mark) အစား ^ (caret) ကိုလည်း သုံးလို့ရပါတယ်။

In [47]:

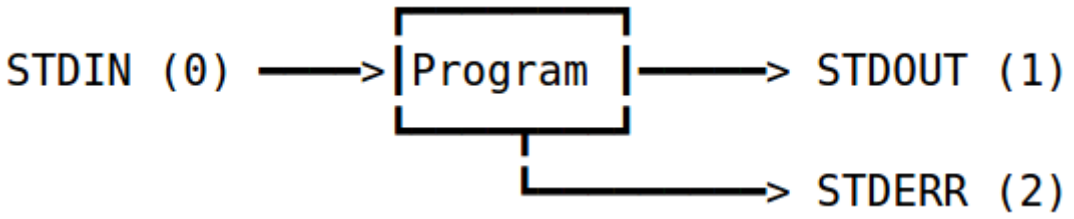
```
ls [^Gg]*
```

```
abstract.txt          conclusion.txt  maik-khe.food.txt  superman
acknowledgements.txt  hello.txt       matrix
city                  m               mba.note.txt
```

အခုလောက်ဆိုရင် wilid card နဲ့ ပတ်သက်ပြီး concept ကတော့ ရသွားလောက်ပါပြီ။

24. Piping and Redirection

ပုံမှန် အားဖြင့် ကျွန်တော်တို့က command တွေကို ကီးဘုတ်ကနေရိုက်ထည့်ပြီး၊ ရလဒ် (output) ကို မော်နီတာမှာ ပြပါတယ်။
Linux/Unix မှာ input/output ကို အောက်ပါအတိုင်း STDIN (0), STDOUT (1), STDERR (2) ဆိုပြီး သတ်မှတ်ထားပါတယ်။



ဒီနေရာမှာ STDIN ရဲ့ default setting (ကျွန်တော်တို့က ဘာမှမညွှန်ကြားရင်) က ကီးဘုတ်ပါ။ ကီးဘုတ်က ဝင်လာတဲ့ command (သို့) input ကို ဖတ်တယ်။
STDOUT ရဲ့default setting က မော်နီတာပါ။ ပရိုဂရမ်ရဲ့ရလဒ် (output) ကို မော်နီတာမှာ ပြပေးတယ်။
STDERR ရဲ့default setting က မော်နီတာပါ။ ပရိုဂရမ် run နေတုန်းမှာ error (အမှာအယွင်းတစ်ခုခု ဖြစ်ခဲ့ရင်) ရှိရင် မော်နီတာမှာ ပြပေးမယ်။

ကျွန်တော်တို့က လိုအပ်တဲ့ အခါမှာ STDIN, STDOUT, STDERR တွေကို ပြောင်းပေးလို့ရပါတယ်။ အဲဒီအလုပ်ကို အင်္ဂလိပ်လို redirection လုပ်တယ်လို့ ခေါ်ပါတယ်။
လက်တွေ့မှာ အဓိကအသုံးများတာက STDOUT, STDERR ရဲ့output တွေကို ဖိုင်တွေမှာ ရေးခိုင်းတာမျိုးပါ။
ဥပမာ ls command ကထွက်လာတဲ့ output ကို ls.out ဆိုတဲ့ ဖိုင်မှာသိမ်းချင်တယ်ဆိုရင် အောက်ပါအတိုင်း ">" (greather than) သင်္ကေတ နဲ့ ညွှန်းပါတယ်။ redirection လုပ်ပါတယ်။

In [48]:

```
ls > ls.out
```

In [49]:

```
cat ls.out
```

```
abstract.txt
acknowledgements.txt
city
conclusion.txt
graph1
Graph1
graph2
Graph2
graph3
Graph3
graph4
Graph4
graph5
hello.txt
ls.out
m
maik-khe.food.txt
matrix
mba.note.txt
superman
```

ls.out ဖိုင်က မရှိသေးရင် ဖိုင်အသစ် ရေးပေးပါတယ်။
တခုသတိထားရမှာက ls.out ဖိုင်က ရှိပြီးသားဆိုရင်၊ အရင်ရှိပြီးသားဖိုင် ကို ထပ်ရေးတာ (overwrite) ဖြစ်လို့၊ အရင် ls.out မှာရှိပြီးသား အချက်အလက်တွေအားလုံး ပျက်သွားမယ် ဆိုတာကိုပါ။

သိထားရမှာက ပရိုဂရမ်ရဲ့ output ကို ဖိုင်မှာ ရေးပေးပါလို့ ညွှန်းပေမဲ့၊ ပုံမှန်အားဖြင့် STDERR ရဲ့ error message ကိုတော့ မော်နီတာဆီ ကို ပို့ပေးပါတယ်။
အဲဒါက ဘာ့ကြောင့်လဲ ဆိုရင် error message ကို ကွန်ပြူတာသုံးနေတဲ့ user က မမြင်ရရင် အဆင်မပြေနိုင်ဘူးလေ။
STDERR ရဲ့ error message ကိုပဲ ဖိုင်မှာ သိမ်းချင်ရင်၊ "2>" ကိုသုံးပြီး redirection လုပ်လို့ရပါတယ်။

Redirection သင်္ကေတတွေအားလုံးကိုတော့ အောက်ပါအတိုင်း ဇယားနဲ့ ပြသထားပါတယ်။

Redirection သင်္ကေတ	အလုပ်လုပ်ပုံ
<	STDIN ကို ကီးဘုတ်ကနေမဟုတ်ပဲ၊ ဖိုင်ကနေ ပေးဖို့အတွက် အသုံးပြုပါတယ်။
>	STDOUT ကို ဖိုင်မှာရေးပေးဖို့အတွက် အသုံးပြုပါတယ်။
>>	STDOUT ကို ရှိနေပြီးသား ဖိုင်ရဲ့နောက်ဆုံးကနေဝင်ရေးပေးဖို့ အသုံးပြုပါတယ်။
>&	STDERR ကနေပေးတဲ့ error message ကိုလည်း ဖိုင်မှာရေးပေးစေချင်ရင် အသုံးပြုပါတယ်။
>>&	>& redirection ရဲ့ >> ပါ။ ရှိပြီးသားဖိုင်ရဲ့နောက်ဆုံးမှာ ဝင်ရေးပေးဖို့ အသုံးပြုပါတယ်။
2>	ပရိုဂရမ်ရဲ့ output မပါပဲ၊ STDERR ရဲ့ error message ကိုပဲ ဖိုင်မှာရေးပေးစေချင်တဲ့ အခါ အသုံးပြုပါတယ်။

Redirection အသုံးပြုပုံကို ဥပမာ တချို့နဲ့ ရှင်းပြပါမယ်။
ls $ ဆိုပြီး run ကြည့်ပါ။
$ ဆိုတဲ့ ဖိုင်၊ ဖိုလ်ဒါကို ရှာမတွေ့ဘူးဆိုပြီးတော့ error message ကို မော်နီတာမှာ ရိုက်ပြပါလိမ့်မယ်။

In [51]:

```
ls $
```

```
ls: cannot access '$': No such file or directory
```

In [ ]:

```
sort > sorted-names <<EOF
> Kyaw Kyaw Moe
> Zaw Zaw
> Mg Mg
> Hla Hla
> Mya Mya
> Blue Ma
> Toe Toe
> Aye Mya
> EOF
```

In [4]:

```
cat sorted-names
```

```
Aye Mya
Blue Ma
Hla Hla
Kyaw Kyaw Moe
Mg Mg
Mya Mya
Toe Toe
Zaw Zaw
```

Piping ဆိုတာကတော့ ပရိဂရမ်တခုရဲ့ output ကို နောက် ပရိဂရမ်တခုကို လက်ဆင့်ကမ်းတာကို ဆိုလိုတာပါ။
"|" (pipe character) ကို သုံးပါတယ်။ ရေတွေကို ရေပိုက်တခုနဲ့ တစ်နေရာကနေ၊ အခြားတစ်နေရာကို ပို့ပေးတဲ့ ပုံစံမျိုး အလုပ်လုပ်ပါတယ်။

ဥပမာ /dev ဖိုလ်ဒါအောက်မှာ ရှိတဲ့ device နဲ့ပတ်သက်တဲ့ ဖိုင်တွေကို ls command ကိုသုံး -l option နဲ့ တစ်ကြောင်းချင်းစီရိုက်ခိုင်းပြီး ထွက်လာတဲ့ output စာကြောင်းတွေကို "|" နဲ့ wc command ဆီကို လက်ဆင့်ကမ်းပြီး စာကြောင်းဘယ်လောက်ရှိသလဲ ရေတွက်ခိုင်းတာပါ။

In [5]:

```
ls -1 /dev | wc
```

```
    213     213    1354
```

အထက်ပါ wc command ကထွက်လာတဲ့ output ကနေ /dev ဖိုလ်ဒါအောက်မှာ device နဲ့ ပတ်သက်တဲ့ဖိုင် စုစုပေါင်း ၂၁၃ ဖိုင် ရှိတယ်ဆိုတာကို သိနိုင်ပါတယ်။
(ဒီနံပါတ်အရေအတွက် က ကွန်ပြူတာတစ်လုံးနဲ့ တစ်လုံး တူမှာ မဟုတ်ပါဘူး)

ဒီလို piping မလုပ်ပဲ၊ ls command ကနေထွက်လာတဲ့ ရလဒ်ကို ဖိုင်တစ်ဖိုင်မှာ သိမ်း၊ ပြီးတော့မှာ အဲဒီဖိုင်ကို wc command ကို သုံးပြီး ဘယ်နှစ်ဖိုင်ရှိသလဲဆိုတာကို ရေတွက်လို့လည်းရပါတယ်။ ဒါပေမဲ့ ဖိုင်အနေနဲ့ သိမ်းထားဖို့ မလိုအပ်ရင် memory ပေါ်မှာပဲ piping လုပ်ပြီး run ရင်ပိုပြီးတော့ မြန်ဆန်ပါလိမ့်မယ်။

In [6]:

```
echo -e "Yangon \nMandalay \nNaypyidaw \nMawlamyine \nBago \nPathein \nPyay \nMonyw
```

အထက်ပါ command က echo နဲ့ မြန်မာနိုင်ငံရဲ့မြို့နာမည်တွေကို ရိုက်ထုတ်ခိုင်းပြီး ထွက်လာတဲ့ output ကို sort command ကိုလက်ဆင့်ကမ်းပြီး အက္ခရာစဉ်လိုက် စီခိုင်းပါတယ်။
sort command က စီပေးလိုက်တဲ့ output ကို redirection သင်္ကေတ တစ်ခုဖြစ်တဲ့ > (greather than) ကိုသုံးပြီး city ဆိုတဲ့ ဖိုင်မှာ သိမ်းခိုင်းပါတယ်။
city ဖိုင်ထဲမှာ ဘယ်လို ရှိနေတယ်ဆိုတာကို အောက်ပါအတိုင်း cat command နဲ့ ရိုက်ကြည့်ပါမယ်။

In [7]:

```
cat city
```

```
Bago
Mandalay
Mawlamyine
Meiktila
Monywa
Myeik
Naypyidaw
Pathein
Pyay
Sittwe
Taunggyi
Yangon
```

နောက်ထပ် ဥပမာတစ်ခုအနေနဲ့ ရှေ့မှာလေ့လာခဲ့တဲ့ "head" နဲ့ "tail" command နှစ်ခုကို pipe နဲ့ပေါင်းသုံးတာကို ပြပါမယ်။ ကျွန်တော်တို့ မြန်မာစာကြောင်းတွေရဲ့ POS tag တွေချည်းပဲသိမ်းထားတဲ့ otest.tag ဖိုင်မှ စာကြောင်း ၂၀၁ ကနေ ၂၀၅ အထိ ရိုက်ထုတ်ခိုင်းမယ်ဆိုရင် အောက်ပါအတိုင်း command ပေးလို့ရပါတယ်။

In [2]:

```
head -n 205 ./otest.tag | tail -n 5
```

```
n adj n part v part part punc
n n ppm pron ppm n v ppm punc
v part part pron ppm v part punc
n n n n ppm n v n n ppm adj v n part v conj n adv v ppm punc
conj n n ppm n conj v part v conj v ppm punc
```

piping က command နှစ်ခုအကြားမှာပဲ လုပ်တာ မဟုတ်ပါဘူး။ command တွေ အများကြီးကို တစ်ခါတည်း piping လုပ်ပြီး သုံးလို့ရပါတယ်။ အောက်ပါ ဥပမာက

In [8]:

```
cat ./otest.tag | tr ' ' '\n' | sort | uniq -c | awk '{ print $2 " " $1}'
```

```
abb 46
adj 608
adv 243
conj 1013
fw 236
int 5
n 5907
num 380
part 4189
part_neg 138
ppm 3431
pron 253
punc 1429
sb 25
tn 211
v 2838
```

26. Regular Expression (RE or regex)

Regular Expression (RE) က အလွယ်ဆုံးရှင်းပြရရင် စာလုံးတွေ၊ စာကြောင်းတွေ၊ ဖိုင်တွေ၊ ဘာသာစကားတွေနဲ့ ပတ်သက်ပြီး ရှာဖွေတဲ့ နေရာ၊ ဝင်ပြင်တဲ့နေရာတွေမှာ အင်မတန်ကို အရေးပါတဲ့ သတ်မှတ်ချက်တစ်ခု၊ ဘာသာစကားတစ်ခုပါ။

အင်္ဂလိပ်လို ပြောရင်တော့ text processing (သို့) pattern recognition လို့ခေါ်ပါတယ်။ အဲဒီ text processing, pattern recognition အလုပ်တွေမှာ RE က အဓိကကျတဲ့ အခန်းကဏ္ဍကပါဝင်ပါတယ်။ ကွန်ပြူတာ application တွေမှာလဲ နေရာပေါင်းစုံမှာကို အသုံးပြုပါတယ်။ ဥပမာ အနေနဲ့ word processor တွေကနေ database application, search engine စသည်ဖြင့်၊ စသည်ဖြင့်။ ဒါ့အပြင် RE က programming languages (eg. Java and JavaScript, Visual Basic and VBScript, C, C++, C#, Perl, Python, Ruby, PHP, elisp, sed, awk etc.) တွေရဲ့အစိတ်အပိုင်း တစ်ပိုင်းအဖြစ်ပါဝင်ပြီး ကောင်းကောင်း အသုံးချတတ်ရင်၊ တကယ့်ကို powerful tool တစ်ခုပါ။

RE ကို တကယ်သိဖို့၊ ကိုယ်လိုချင်တဲ့ စာကြောင်းတွေ၊ စကားလုံးအစိတ်အပိုင်းတွေကို ဆွဲထုတ်ယူနိုင်ဖို့ အထိက အချိန်ယူလေ့လာရပါတယ်။ လက်တွေ့သုံးရင်းနဲ့ လေ့လာရတယ်ဆိုရင် ပိုမှန်ပါတယ်။ ကျွန်တော် ဒီနေရာမှာလည်း RE ဆိုတာက ဘာလဲဆိုတာသိအောင်၊ မိတ်ဆက်အနေနဲ့ အသုံးပြုပုံ အချို့ကို ရှင်းပြရင်း၊ concept ရဖို့ကို ရည်ရွယ်ပါတယ်။ ကျွန်တော်ကိုယ်တိုင်လည်း RE က မသိသေးတာတွေအများကြီးပါ။ တကယ် RE ကိုစိတ်ဝင်စားလို့ သီအိုရီပိုင်းကို ပိုသိချင်ရင် စာအုပ်တွေဖတ်၊ လက်တွေ့ကြုံရတဲ့ text processing ပြဿနာတွေကို အမျိုးမျိုး စမ်းသပ်သုံးကြည့်မှသာ ကျွမ်းကျင်လာပါလိမ့်မယ်။

Characters

သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာ RE	ဥပမာ RE နဲ့ ကိုက်သောရလဒ်
\d	၀ ကနေ ၉ အတွင်းက ဂဏန်းတစ်လုံး	TV_Channel_\d\d	Watching TV_Channel_17.
\D	ဂဏန်းတစ်လုံး မဟုတ်	\D	Give me version 0.9!
\w	စာလုံးတစ်လုံး (သို့) ဂဏန်းတစ်လုံး (သို့) underscore "_"	\w\w\w	This is version 0.9!
\W	\w မဟုတ်သော စာလုံးတစ်လုံး	\W\W\W\W	Do you know +-*%?
\s	whitespace စာလုံး (space, tab, newline, carriage return, vertical tab)	Mg\sMg	Mg Mg and Ma Ma!
\S	whitespace မဟုတ်တဲ့ စာလုံးတစ်လုံး	\S\S\S\S	I love Aikido!

သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာ RE	ဥပမာ RE နဲ့ ကိုက်သောရလဒ်
.	ဘာစာလုံးမဆို (သို့သော် စာကြောင်းဖြတ်တာနဲ့ မပတ်သတ်သော စာလုံး)	L..	Working at AI Lab.
\	အထူးစာလုံး (ဥပမာ RE ကယူသုံးထားသောစာလုံးများ)တွေကို Escape လုပ်တဲ့အခါသုံးသော သင်္ကေတ	\.*\?\^$\{\(	.*?^${(

Quantifiers

အရေအတွက် သတ်မှတ်ချက် သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာRE	ဥပမာ RE နဲ့ ကိုက်သောရလဒ်
+	တစ်ခု (သို့) တစ်ခုထက်ပို	x+	xx (or) xxxxx (or) xxxxxxxxxx
*	သုည (သို့) တစ်ခုထက်ပို	xyz*	xyz (or) xxyzzzzz (or) xxxyyyzzz
?	သုည (သို့) တစ်ခု	x?	x (or) xx
{3}	သုံးခါတိတိ	\w{3}	abc (or) xyz (or) aaa
{2,5}	နှစ်ခါ ကနေ ၅ခါအတွင်း	\d{2,5}	111 (or) 1223 (or) 123456
{2,}	နှစ်ခါ နှင့် အထက်	\w{2,}	pro (or) program (or) programming

Whitespace

သင်္ကေတ	သတ်မှတ်ချက်
\t	Tab စာလုံးကို ကိုယ်စားပြုတဲ့ စာလုံးတစ်လုံး
\r	Carriage return ကို ကိုယ်စားပြုတဲ့ စာလုံးတစ်လုံး
\n	Line feed ကို ကိုယ်စားပြု (Linux OS မှာစာကြောင်း အောက်တစ်ကြောင်းဆင်းဖို့အတွက်သုံး) တဲ့စာလုံး တစ်လုံး
\r\n	Windows OS မှာသုံးတဲ့ Line feed စာလုံး
\h	horizontal whitespace စာလုံးတစ်လုံး (e.g. Tab, space)
\b	backspace စာလုံး တစ်လုံး
\v	vertical whitespace စာလုံး တစ်လုံး (ဥပမာ line feed, carriage return, vertical tab, form feed etc.)
\N	line break မဟုတ်တဲ့ စာလုံး တစ်လုံး
\H	horizontal whitespace မဟုတ်တဲ့ စာလုံး တစ်လုံး
\V	vertical whitespace မဟုတ်တဲ့ စာလုံး တစ်လုံး

Character Classes

သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာ RE	ဥပမာ RE နဲ့ ကိုက်သောရလဒ်
[...]	လေးထောင့်ကွင်းနှစ်ခုအတွင်းက စာလုံးတစ်လုံး	[ABC]	A (or) B (or) C
[x-y]	x ကနေ y ကြားထဲမှာ ရှိသော စာလုံးတစ်လုံး	[A-E]	A (or) B (or) C (or) D (or) E
[^]	လေးထောင့်ကွင်း နှစ်ခုအတွင်းမှာ ရှိနေတဲ့ စာလုံးတွေထဲက မဟုတ်တဲ့ စာလုံး	[^xyz]	a (or) b (or) k

သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာ RE	ဥပမာ RE နဲ့ ကိုက်သောရလဒ်
[^x-y]	x ကနေ y ကြားထဲမှာ ရှိမနေသော စာလုံး တစ်လုံး	[^0-7]	8 (or) 9
[\xnn]	hexadecimal နံပါတ် nn နဲ့ညီမျှတဲ့ စာလုံး တစ်လုံး	[\x41\x42\x43]	A (or) B (or) C
[\uFFFF]	Unicode နံပါတ် FFFF နဲ့ ညီမျှတဲ့ စာလုံး တစ်လုံး	[\u1000]	က

Anchors and Boundaries

သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာ RE	ဥပမာ RE နဲ့ ကိုက်သောရလဒ်
^	စာကြောင်း တစ်ကြောင်းရဲ့အစ (ဒီနေရာမှာ စာကြောင်းတစ်ကြောင်းဆိုတာက multiline mode ပါ ပါဝင်တယ်)	^This	This and that.
$	စာကြောင်း တစ်ကြောင်းရဲ့အဆုံး (ဒီနေရာမှာ စာကြောင်း တစ်ကြောင်းဆိုတာက multiline mode ပါ ပါဝင်တယ်)	တယ်$	စာကြိုးစားတယ်
\b	စာလုံးတစ်လုံးရဲ့အဆုံး (RE အင်ဂျင်အများစုက ASCII စာလုံးတွေကို အခြေခံထားတာဖြစ်လို့ မြန်မာစာအတွက် က အဆင်မပြေတာတွေရှိပါတယ်။) နောက်တစ်ချက် က ဒီ \b က character class အနေနဲ့ သုံးတဲ့ အခါမှာတော့ backspace ဖြစ်ပါတယ်။	\ba\w*\b	Who am I?
\B	စာလုံးတစ်လုံးရဲ့အဆုံးမဟုတ်	\Bo\B	quick brown fox jump over the lazy dog

POSIX Classes

သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာ RE	ဥပမာ RE နဲ့ ကိုက်သောရလဒ်
[:alpha:]	ASCII စာလုံး A-Z နှင့် a-z	[[:alpha:]]	Happy Departure!
[:alnum:]	ASCII ဂဏန်းနံပါတ်၊ A-Z နှင့် a-z	[[:alnum:]]{6}	Happy Birthday to you!
[:punct:]	ASCII ပုဒ်ဖြတ်သင်္ကေတ (punctuation) စာလုံး	[[:punct:]]+	Hi! Nay Kaung Lar?

POSIX classes တွေက လည်း အများကြီးရှိပါတယ်။ ဒီနေရာမှာ တစ်ခု သိထားသင့်တာက POSIX classes တွေနဲ့ တချို့escape sequence တွေက အတူတူပဲ ဆိုတာကိုပါ။ အောက်ပါဇယားမှာ ပြထားတဲ့ ဥပမာ တချို့ကို မှီငြမ်းပါ။

Escape sequence သင်္ကေတ	POSIX classes
\d	[[:digit:]]
\l	[[:lower:]]
\s	[[:space:]]
\u	[[:upper:]]
\w	[[:word:]]
\D	[^[:digit:]]
\L	[^[:lower:]]
\S	[^[:space:]]
\U	[^[:upper:]]

Escape sequence သင်္ကေတ	POSIX classes
\W	[^[:word:]]

Unicode Regular Expression

ယူနီကုဒ်နဲ့ ပတ်သက်တဲ့ RE ကိုလည်း နည်းနည်း မိတ်ဆက်ပါမယ်။ တိုတိုပြောရရင် အင်္ဂလိပ်စာအတွက် RE token အားလုံးက အလုပ်လုပ်ပါတယ်။ တချို့RE တွေက မြန်မာစာ အပါအဝင် ဘာသာစကားအားလုံးအတွက် အမြဲတမ်းအဆင်ပြေတာမဟုတ်ပါဘူး။ အဲဒီလိုအခါမျိုးမှာ Unicode RE ရဲ့အကူအညီကို လှမ်းယူတာ မျိုးလည်း လုပ်ရတတ်ပါတယ်။ ယူနီကုဒ် RE ဆိုတာက ယူနီကုဒ်အဖွဲ့အစည်းက သတ်မှတ်ထားတဲ့ သတ်မှတ်ချက်တွေကို၊ RE အနေနဲ့လည်း အလုပ်လုပ်လို့ရအောင် သင်္ကေတတွေ သတ်မှတ်ထားတာလို့ လွယ်လွယ်ကူကူမှတ်လို့ ရပါတယ်။ Unicode RE ကလည်း တကယ့်ကို အများကြီးပါ။ အားလုံးကို မှတ်ထားစရာ မလိုပါဘူး။ သဘောတရားရအောင်သာ Unicode RE တချို့က အောက်ပါဇယားအတိုင်းပါ။

Unicode RE သင်္ကေတ	သတ်မှတ်ချက်
\X	ယူနီကုဒ် အတွက် RE "." ပါ
\x{FFFF}	ယူနီကုဒ်စာလုံးတွေကို ကုဒ်နံပါတ်နဲ့ ရိုက်နိုင်ပါတယ်။ ဥပမာ \x{1F637} ဆိုရင် 😷 (FACE WITH MEDICAL MASK) ကို ရိုက်ပေးပါလိမ့်မယ်။
\x{1000}{10}	မြန်မာစာ စာလုံး "က" ကို ၁၀ခါ
\p{L} (သို့) \p{Letter}	Unicode စာလုံး တစ်လုံး
\p{Ll} (သို့) \p{Lowercase_Letter}	အင်္ဂလိပ်ကဲ့သို့ စာလုံးအကြီးအသေး ရှိတဲ့ ဘာသာစကားတွေရဲ့စာလုံးအသေး တစ်လုံး
\p{Lu} (သို့) \p{Uppercase_Letter}	အင်္ဂလိပ်ကဲ့သို့ စာလုံးအကြီးအသေး ရှိတဲ့ ဘာသာစကားတွေရဲ့စာလုံးအကြီး တစ်လုံး
\p{S} (သို့) \p{Symbol}	သင်္ချာ သင်္ကေတ၊ ငွေကြေး သင်္ကေတ၊ အလွယ်တကူပုံတွေကို ဆွဲတဲ့နေရာမှာ သုံးတဲ့ စသည့်သင်္ကေတများထဲက တစ်ခု
\p{Sm} (သို့) \p{Math_Symbol}	သင်္ချာ သင်္ကေတ တစ်ခု
\p{N} (သို့) \p{Number}	ဘာသာစကားမျိုးစုံရဲ့ဂဏန်းနံပါတ်များထဲက ဂဏန်းတစ်ခု
\p{Nl} (သို့) \p{Letter_Number}	စာလုံးကဲ့သို့သော နံပါတ်များ။ ဥပမာ ရောမနံပါတ်များ ဖြစ်ကြတဲ့ I, V, IX စသည်ဖြင့်
\p{Myanmar}	ယူနီကုဒ်အဖွဲ့အစည်းက သတ်မှတ်ထားပြီးဖြစ်တဲ့ မြန်မာယူနီကုဒ်မှာပါဝင်တဲ့ စာလုံးများထဲက စာလုံးတစ်လုံး
\p{InMyanmar}	ယူနီကုဒ်အဖွဲ့အစည်းက သတ်မှတ်ထားပြီးဖြစ်တဲ့ မြန်မာယူနီကုဒ်နံပါတ် U1000 ကနေ U109F အကြားဝင်တဲ့ စာလုံးများ

Inline Modifier

သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာ RE	ဥပမာ RE နဲ့ ကိုက်သောရလဒ်
(?i)	အင်္ဂလိပ်စာ စာလုံးအကြီး၊ အသေးကို မခွဲခြားပဲ၊ အတူတူပဲဆိုတဲ့ သတ်မှတ်ချက် (case insensitive)	(?i)phone	iPhone is a mobile phone. Yes, it is a SMART PHONE!
(?c)	အင်္ဂလိပ်စာ စာလုံးအကြီး၊ အသေးကို သပ်သပ်စီခွဲထားတဲ့ သတ်မှတ်ချက် (case sensitive)	(?c)phone	iPhone is a mobile phone. Yes, it is a SMART PHONE!

သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာ RE	ဥပမာ RE နဲ့ ကိုက်သောရလဒ်
(?x)	RE တွေကို ဖတ်ရလွယ်ကူဖို့အတွက်၊ RE ရဲ့ token တစ်ခုနဲ့ တစ်ခု အကြားမှာ ရှိတဲ့ space တွေကို လစ်လျူရှုဖို့အတွက် (free-spacing mode လို့ခေါ်) သုံးတဲ့ inline modifier ပါ။ comment mode, whitespace mode လို့လည်း သိကြပါတယ်။	(?x)a b c	abc and a b c and ABC and abcabc
(?t)	free-spacing mode ကို ပိတ်ချင်တဲ့အခါမှာ အသုံးပြုပါတယ်။ Tcl programming language မှာပဲသုံးနိုင်။	(?t)a b c	abc and a b c and ABC and abcabc
(?s)	single line mode ကို active ဖြစ်ဖို့အတွက် အသုံးပြုပါတယ်။ single line mode မှာတော့ RE "." က line break စာလုံးတွေအပါအဝင်၊ စာလုံးအားလုံး ကို ကိုယ်စားပြုပါလိမ့်မယ်။ "dot all mode" ဆိုပြီး ခေါ်ကြပါတယ်။	(?s)(^.*\d+$)	Line no.1\nLine no.2\nLine no.3
(?m)	multi-line mode ကို active ဖြစ်ဖို့အတွက် အသုံးပြုပါတယ်။	(?m)(^.*\d+$)	Line no.1\nLine no.2\nLine no.3

Lookahead and Lookbehind

Lookaround သင်္ကေတ	အလုပ်လုပ်ပုံ	ဥပမာ RE	ဥပမာ RE နဲ့ ကိုက်သော ရလဒ်
(?=regex)	Positive lookahead လို့ခေါ်ပါတယ်။ ရှာဖွေမဲ့ စာလုံးရဲ့အရှေ့မှာ (ညာဘက်) regex pattern ရှိနေတဲ့ အခြေအနေ	.{3}(?=\d{5})	abc12345
(?!regex)	Negative lookahead လို့ခေါ်ပါတယ်။ ရှာဖွေမဲ့ စာလုံးရဲ့အရှေ့မှာ (ညာဘက်) regex pattern မရှိတဲ့ အခြေအနေ	Hard(?!Rock)	HardRock HardMusic
(?<=regex)	Positive lookbehind လို့ခေါ်ပါတယ်။ ရှာဖွေမဲ့ စာလုံးရဲ့အနောက်မှာ (ဘယ်ဘက်) regex pattern ရှိနေတဲ့ အခြေအနေ	(?<=\d)D	5D fiveD oneD 1D
(?<!regex)	Negative lookbehind လို့ခေါ်ပါတယ်။ ရှာဖွေမဲ့ စာလုံးရဲ့အနောက်မှာ (ဘယ်ဘက်) regex pattern မရှိတဲ့ အခြေအနေ	(?<!\d)D	5D fiveD oneD 1D

Lookahead, Lookbehind ကို command prompt မှာ grep (ကိုယ်လိုချင်တဲ့ RE နဲ့ ရှာဖွေပေးနိုင်တဲ့ ပရိုဂရမ်) commandနဲ့ အောက်ပါအတိုင်း စမ်းrun ကြည့်ပါ။

In [16]:

```
echo -e "TuuWarWarTuu" | grep -P 'War(?=War)'
```

Tuu**War**WarTuu

In [17]:

```
echo -e "TuuWarWarTuu" | grep -P 'War(?!War)'
```

TuuWar**War**Tuu

In [18]:

```
echo -e "TuuWarWarTuu" | grep -P '(?<=Tuu)War'
```

Tuu**War**WarTuu

In [19]:

```
echo -e "TuuWarWarTuu" | grep -P '(?<!Tuu)War'
```

TuuWar**War**Tuu

Greedy and Lazy

RE မှာ greedy search နဲ့ lazy search ဆိုပြီး နှစ်မျိုးရှိတယ်။ Greedy search က ပေးလိုက်တဲ့ regex pattern (သို့) condition ကို ပေးလိုက်တဲ့ စာကြောင်း အပေါ်မှာ တစ်ခေါက်တွေ့တာနဲ့ ရပ်မသွားပဲ ထပ်ခါထပ်ခါ ဆက်ရှာပြီး၊ နောက်ဆုံး အကြိမ် ပြေလည်သည်အထိ ရှာတဲ့ပုံစံပါ။ Lazy search က regex pattern ကို တစ်ခေါက်တွေ့တာနဲ့ ထပ်မရှာတော့ပဲ ရပ်သွားပါလိမ့်မယ်။ ပုံမှန်အားဖြင့် RE က greedy search ပါ။ RE engine ကို Lazy search အနေနဲ့ အလုပ်လုပ်ပေးစေချင်ရင်တော့ condition (သို့) regex pattern ရဲ့နောက်မှာ "?" ကိုထည့်ပါ။

Greedy ကနေ Lazy အဖြစ်ပြောင်းတဲ့ ပုံစံတချို့ကို လေ့လာဖို့အတွက် အောက်ပါဇယားကို မြင်ပါ။

Greedy	Lazy	အလုပ်လုပ်ပုံ
*	*?	သူည (သို့) တစ်ခုထက်ပို
+	+?	တစ်ခု (သို့) တစ်ခုထက်ပို
?	??	သူည (သို့) တစ်ခု
{n}	{n}?	n အကြိမ်တိတိ
{n,}	{n,}?	အနည်းဆုံး n အကြိမ်နဲ့ညီမျှ
{n,m}	{n,m}?	n အကြိမ် ကနေ m အကြိမ်အထိ

In [20]:

```
echo -e "<em>Hello World</em>" | grep -P '<.+>'
```

Hello World

In [21]:

```
echo -e "<em>Hello World</em>" | grep -P '<.+?>'
```

****Hello World****

In [22]:

```
echo -e "puppy" | grep -P 'p.+p'
```

puppy

In [23]:

```
echo -e "puppy" | grep -P 'p.+?p'
```

puppy

ယခုလောက်ဆိုရင် RE ရဲ့ concept နှင့်တကွ အသုံးပြုပုံပြုနည်း သဘောတရားကိုတော့ ရလောက်ပြီလို့ယူဆပါတယ်။ မြင်တတ်တဲ့ သူဆို လျှင် wild card နဲ့ နှိုင်းယှဉ်ကြည့်ပြီး၊ wild card ရဲ့ "?" က RE ရဲ့ "." နဲ့ သွားတူတာ။ ထိုနည်းလည်းကောင်း wild card ရဲ့ "*" က RE ရဲ့ ".*" နဲ့ တူပုံကိုလည်း သိနေလောက်ပါပြီ။

RE ကို linuxပရိဂရမ် အမျိုးမျိုးမှာ လိုအပ်ချက်နဲ့အညီ ပြင်ဆင်ပြီးသုံးကြလို့ သူရဲ့ syntax (တွဲသုံးတဲ့ ပုံစံ)က အမျိုးမျိုးရှိပါတယ်။ ဥပမာ တချို့မှာ အောက်ပါအတိုင်း ဖြစ်ပါသည်။

*awk regular expression syntax, egrep regular expression syntax, ed regular expression syntax, emacs regular expression syntax, gnu-awk regular expression syntax, grep regular expression syntax

လက်တွေ့မှာ ရာနဲ့ထောင်နဲ့ချီရှိတဲ့ ဖိုင်တွေနဲ့ အလုပ်လုပ်ရတော့မယ်ဆိုရင် wild card တွေ၊ regular expression တွေကို သုံးပြီး အလုပ်လုပ်မှ သက်သာပါလိမ့်မယ်။ သို့သော် တကယ့်လက်တွေ့ text processing ရဲ့ပြဿနာတွေကို RE ကို သုံးပြီး ဖြေရှင်းနိုင်ဖို့ကတော့ လက်တွေ့စမ်းသုံးမှသာ ကျွမ်းကျင်လာပါလိမ့်မယ်။ ပထမ အစပိုင်းမှာတော့ ကိုယ်လိုချင်တဲ့ text pattern ကို တော်တော်နဲ့ ဆွဲယူလို့မရလို့ စိတ်ပင်ပန်းရတာတွေလည်း ရှိမှာ မလွဲမသွေပါပဲ။ စမ်းသုံးကြည့်ရင်း၊ စမ်းသုံးကြည့်ရင်း နဲ့ RE နဲ့ အလုပ် လုပ်လို့ရလာပြီဆိုရင်တော့၊ RE ရဲ့ ပါဝါကိုလည်း သိရှိပြီး၊ ခက်ခဲတဲ့ ပြဿနာတွေကိုလည်း ဖြေရှင်းလာနိုင်ပါလိမ့်မယ်။

ဒီအထိ linux command တွေရော wild card နဲ့ RE တွေကို ရိုက်စမ်းကြည့်ရင်း၊ သဘောပေါက်အောင် သေသေချာချာ ကြိုးစားပြီး လေ့လာခဲ့တဲ့သူ ဆိုရင်တော့ တော်တော်လေး ခရီးပေါက်နေပါပြီ။ အောက်ပါ RE နဲ့ ပတ်သက်ပြီး လူသိများတဲ့ ကာတွန်းကို နောက်ဆုံးအနေနဲ့ message ပေးရင်း၊ RE ရဲ့သင်ကြားမှုကို ရပ်လိုက်တော့မယ်။

Above RE Comic is from: xkcd, A webcomic of romance, sarcasm, math, and language. (https://xkcd.com/208/)

27. find (search for files in a directory hierarchy)

ဖိုင်ကို ဖိုလ်ဒါအောက် အဆင့်ဆင့် ရှာဖွေတဲ့ နေရာမှာ အသုံးပြုပါတယ်။

ဘာ parameter မှမပေးပဲ find command တစ်ခုတည်းကိုပဲ ရိုက်ရင် လက်ရှိရောက်နေတဲ့ ဖိုလ်ဒါအောက်မှာ ရှိတဲ့ ဖိုင်၊ ဖိုလ်ဒါ၊ ဖိုလ်ဒါတွေ အောက်မှာ ရှိတဲ့ ဖိုင်တွေ အားလုံးကို ရှာဖွေပြီး ရိုက်ထုတ်ပြပါလိမ့်မယ်။

In [1]:

```
find
```

```
.
./folder-new
./folder-new/folder-new
./city
./screen
./screen/KaKhaGa-output.png
./screen/echo-kagyi.png
./screen/redirections.png
./screen/bayesian-formula.png
./screen/regular_expressions.png
./screen/less-cmd-output.png
./linux-commands.ipynb
./.fileC
./folder-athit
./data1
./all-vi-info.txt
./folderB
./folderB/127-135.pdf
./folderB/fileA
./folderB/otest
./data4
./fmt.out
./echo
./.ipynb_checkpoints
./.ipynb_checkpoints/linux-commands-checkpoint.ipynb
./my-news
./folderA
./folderA/127-135.pdf
./ls.out
./news
./fileA
./data3
./otest.word
./otest
./item-list
./data5
./data2
./fileB
./linux-commands.ipynb.bak
./wildcard
./wildcard/graph2
./wildcard/Graph4
./wildcard/superman
./wildcard/hello.txt
./wildcard/abstract.txt
./wildcard/graph4
./wildcard/city
./wildcard/Graph3
./wildcard/maik-khe.food.txt
./wildcard/Graph1
./wildcard/mba.note.txt
./wildcard/matrix
./wildcard/ls.out
./wildcard/graph1
./wildcard/acknowledgements.txt
./wildcard/m
./wildcard/graph3
```

```
./wildcard/.my-private-info.txt

./wildcard/graph5
./wildcard/conclusion.txt
./wildcard/Graph2
./otest.tag
./sorted-names
./newfile
```

ဖိုလ်ဒါနာမည် (သို့) ရာဖွေချင်တဲ့ path လမ်းကြောင်းကို ပေးရင်တော့၊ အဲဒီပေးလိုက်တဲ့ path အောက်မှာရှိတဲ့ ဖိုင်၊ ဖိုလ်ဒါတွေရဲ့နာမည် အားလုံးကို ရိုက်ထုတ်ပြပါလိမ့်မယ်။

In [24]:

```
find ./folderA
```

```
./folderA
./folderA/127-135.pdf
```

ဖိုင်ကို နာမည်နဲ့တကွ ရာဖွေချင်ရင်တော့ -name option ကိုသုံးပါတယ်။

In [25]:

```
find -name 127-135.pdf
```

```
./folderB/127-135.pdf
./folderA/127-135.pdf
```

ဖိုလ်ဒါတွေရဲ့အတွင်းပိုင်းကို ဘယ်နှစ်ဆင့်အထိပဲ ဝင်ရာပေးပါဆိုတာကိုလည်း -maxdepth ဆိုတဲ့ option နဲ့ ကန့်သတ်ပေးနိုင်ပါတယ်။

-maxdepth 3 option ကို တပ်ပြီး ဖိုလ်ဒါတွေထဲဝင်ရာတဲ့ အခါမှာ အောက်ဘက် သုံးဆင့် အထိပဲ ဝင်ရာပေးပါဆိုပြီး ကန့်သတ်ပေးတဲ့ ပုံစံက အောက်ပါအတိုင်းပါ။

In [6]:

```
find ~ -maxdepth 3 -name "127-135.pdf"
```

```
find: ‘/home/lar/.dbus’: Permission denied
find: ‘/home/lar/.cache/dconf’: Permission denied
/home/lar/bk-dlbox/tool4all/127-135.pdf
/home/lar/linux-cmd/folderB/127-135.pdf
/home/lar/linux-cmd/folderA/127-135.pdf
```

အထက်မှာ မြင်ရတဲ့ permission denied ဆိုတဲ့ message က ဖော်ပြပါဖိုလ်ဒါအောက်ကို ဝင်ခွင့်မရှိလို့ ပေးတဲ့ error message ဖြစ်ပါတယ်။

-maxdepth နဲ့ ဆန့်ကျင်ဘက် -mindepth ဆိုတာလည်းရှိပါတယ်။
သူကတော့ find command က စမရာခင်မှာ ဖိုလ်ဒါရဲ့အောက်ဖက်ကို n အဆင့်အရင်ဆုံးဆင်းခိုင်းတာပါ။
ဥပမာ -mindepth 3 လို့ option ပေးခဲ့ရင် ရာခိုင်းတဲ့ ဖိုင် (သို့) ဖိုလ်ဒါတွေကို မရာခင်မှာ အရင်ဆုံး ဖိုလ်ဒါ သုံးဆင့်ထဲဝင်ပြီးမှ ရာပေးပါလို့ ခိုင်းတာပါ။

-mindepth နဲ့ -maxdepth option နှစ်ခုကိုတွဲပြီးတော့ လည်းသုံးနိုင်ပါတယ်။

ဥပမာ အောက်ပါ command က ဖိုလ်ဒါ depth အောက်ဘက် နှစ်ဆင့် ကနေ ငါးဆင့် အကြားမှာရှိတဲ့ ".txt" extension နဲ့ ဆုံးတဲ့ ဖိုင်နာမည်တွေကိုပဲ ရာပေးပါဆိုပြီးခိုင်းတာပါ။

```
find -mindepth 2 -maxdepth 5 -name "*.txt"
```

`-not option` ကိုပေးပြီး ဒီဖိုင်ကိုတော့ မရှာပါနဲ့ ဆိုပြီးလည်း ကန့်သတ်လို့လည်း ရပါတယ်။

အောက်ပါ `find command` ဥပမာက `~/linux-cmd/` ဖိုလ်ဒါအောက်မှာဝင်ရှာပေးပါ။
ဒါပေမဲ့ `d` နဲ့စတဲ့ ဖိုင်နာမည်တွေကိုတော့ မရှာပါနဲ့လို့ ပြောပြီး ခိုင်းတာပါ။

In [7]:

```
find ~/linux-cmd/ -not -name "d*"
```

```
/home/lar/linux-cmd/
/home/lar/linux-cmd/folder-new
/home/lar/linux-cmd/folder-new/folder-new
/home/lar/linux-cmd/city
/home/lar/linux-cmd/screen
/home/lar/linux-cmd/screen/KaKhaGa-output.png
/home/lar/linux-cmd/screen/echo-kagyi.png
/home/lar/linux-cmd/screen/redirections.png
/home/lar/linux-cmd/screen/bayesian-formula.png
/home/lar/linux-cmd/screen/regular_expressions.png
/home/lar/linux-cmd/screen/less-cmd-output.png
/home/lar/linux-cmd/linux-commands.ipynb
/home/lar/linux-cmd/.fileC
/home/lar/linux-cmd/folder-athit
/home/lar/linux-cmd/all-vi-info.txt
/home/lar/linux-cmd/folderB
/home/lar/linux-cmd/folderB/127-135.pdf
/home/lar/linux-cmd/folderB/fileA
/home/lar/linux-cmd/folderB/otest
/home/lar/linux-cmd/fmt.out
/home/lar/linux-cmd/echo
/home/lar/linux-cmd/.ipynb_checkpoints
/home/lar/linux-cmd/.ipynb_checkpoints/linux-commands-checkpoint.ipynb
/home/lar/linux-cmd/my-news
/home/lar/linux-cmd/folderA
/home/lar/linux-cmd/folderA/127-135.pdf
/home/lar/linux-cmd/ls.out
/home/lar/linux-cmd/news
/home/lar/linux-cmd/fileA
/home/lar/linux-cmd/otest.word
/home/lar/linux-cmd/otest
/home/lar/linux-cmd/item-list
/home/lar/linux-cmd/fileB
/home/lar/linux-cmd/linux-commands.ipynb.bak
/home/lar/linux-cmd/wildcard
/home/lar/linux-cmd/wildcard/graph2
/home/lar/linux-cmd/wildcard/Graph4
/home/lar/linux-cmd/wildcard/superman
/home/lar/linux-cmd/wildcard/hello.txt
/home/lar/linux-cmd/wildcard/abstract.txt
/home/lar/linux-cmd/wildcard/graph4
/home/lar/linux-cmd/wildcard/city
/home/lar/linux-cmd/wildcard/Graph3
/home/lar/linux-cmd/wildcard/maik-khe.food.txt
/home/lar/linux-cmd/wildcard/Graph1
/home/lar/linux-cmd/wildcard/mba.note.txt
/home/lar/linux-cmd/wildcard/matrix
/home/lar/linux-cmd/wildcard/ls.out
/home/lar/linux-cmd/wildcard/graph1
/home/lar/linux-cmd/wildcard/acknowledgements.txt
/home/lar/linux-cmd/wildcard/m
/home/lar/linux-cmd/wildcard/graph3
/home/lar/linux-cmd/wildcard/.my-private-info.txt
/home/lar/linux-cmd/wildcard/graph5
/home/lar/linux-cmd/wildcard/conclusion.txt
/home/lar/linux-cmd/wildcard/Graph2
/home/lar/linux-cmd/otest.tag
```

```
/home/lar/linux-cmd/sorted-names
/home/lar/linux-cmd/newfile
```

အထက်ပါ ခဲ့တဲ့ -not အစား "!" ကိုလည်း သုံးနိုင်ပါတယ်။
ဥပမာ find ~/linux-cmd/ ! -name "d*"

find command ရဲ့ကောင်းတဲ့အချက် နောက်တစ်ချက်က ဖိုင်တွေကိုရှာဖွေခိုင်းတဲ့ အခါမှာ search criteria တစ်မျိုးထက်မကကို ပေါင်းပြီး တခါတည်းပေးခိုင်းလို့ရတဲ့ အချက်ပါ။

အောက်ပါ ဥပမာက ဖိုင်နာမည် f နဲ့ စတာကို ရှာပေးပါ။
ဒါပေမဲ့ w နဲ့ စတဲ့ ဖိုင်တွေ၊ screen နဲ့ စတဲ့ ဖိုင်တွေကို မရှာပါနဲ့၊ မလိုချင်ဘူးလို့ ကန့်သတ်တာပါ။

In [17]:

```
find ~/linux-cmd/ -name "f*" ! -name "w*" ! -name "screen*"
```

```
/home/lar/linux-cmd/folder-new
/home/lar/linux-cmd/folder-new/folder-new
/home/lar/linux-cmd/folder-athit
/home/lar/linux-cmd/folderB
/home/lar/linux-cmd/folderB/fileA
/home/lar/linux-cmd/fmt.out
/home/lar/linux-cmd/folderA
/home/lar/linux-cmd/fileA
/home/lar/linux-cmd/fileB
```

ဖိုင်ကိုပဲ ရှာပေးပါ။ ဖိုလ်ဒါကိုပဲ ရှာပေးပါဆိုပြီး -type option ပေးပြီး ကန့်သတ်လို့ရပါတယ်။

-type f က file ကို ကိုယ်စားပြုပါတယ်။
-type d က directory ကို ကိုယ်စားပြုပါတယ်။

In [19]:

```
find ~/linux-cmd/ -type f -name "*.txt"
```

```
/home/lar/linux-cmd/all-vi-info.txt
/home/lar/linux-cmd/wildcard/hello.txt
/home/lar/linux-cmd/wildcard/abstract.txt
/home/lar/linux-cmd/wildcard/maik-khe.food.txt
/home/lar/linux-cmd/wildcard/mba.note.txt
/home/lar/linux-cmd/wildcard/acknowledgements.txt
/home/lar/linux-cmd/wildcard/.my-private-info.txt
/home/lar/linux-cmd/wildcard/conclusion.txt
```

In [22]:

```
find ~/linux-cmd/ -type d -name "wild*"
```

```
/home/lar/linux-cmd/wildcard
```

find command နဲ့ ဖိုင်တွေကို ရှာဖွေခိုင်းတဲ့ အခါမှာ ပုံမှန်အားဖြင့် က linux ရဲ့အလုပ်လုပ်တဲ့ ပုံစံအတိုင်း စာလုံးအကြီးအသေးကို ခွဲပြီး အလုပ် လုပ်ပါတယ်။ အဲဒါကြောင့် ပေးလိုက်တဲ့ နာမည်က စာလုံးအသေးဆိုရင် အသေးကိုပဲ ရှာပါမည်။ တကယ်လို့ အင်္ဂလိပ်စာ စာလုံးအကြီးအသေး ကို မခွဲချင်ဘူး၊ ကြီးကြီးသေးသေး ဒီနာမည်ပါရင် ပြီးတာပဲ ဆိုတဲ့ အခြေအနေမျိုးကို လိုချင်ရင်တော့ -iname option ကို သုံးပါတယ်။

In [28]:

```
find ~/linux-cmd/ -type d -iname "foldera"
```

```
/home/lar/linux-cmd/folderA
```

တစ်ခုထက်မကတဲ့ ဖိုလ်ဒါအောက်မှာ ရှာခိုင်းချင်တဲ့ အခါမှာ၊ ဝင်ရှာစေချင်တဲ့ ဖိုလ်ဒါနာမည်တွေကို ရှေ့ပိုင်းမှာ တန်းစီပြီး ရိုက်ထည့်ပေးလို့ရပါတယ်။

အောက်ပါ ဥပမာကို ကြည့်ပါ။

ဖိုလ်ဒါ ~/linux-cmd/folderA နဲ့ ဖိုလ်ဒါ ~/linux-cmd/wildcard/ အောက်မှာရှိတဲ့ ဖိုင်တွေထဲမှာမှ "p" ပါတဲ့ ဖိုင်နာမည်တွေကို ရာပေးပါဆိုပြီး အမိန့်ပေးတာပါ။

In [31]:

```
find ~/linux-cmd/folderA/ ~/linux-cmd/wildcard/ -name "*p*"
```

```
/home/lar/linux-cmd/folderA/127-135.pdf
/home/lar/linux-cmd/wildcard/graph2
/home/lar/linux-cmd/wildcard/Graph4
/home/lar/linux-cmd/wildcard/superman
/home/lar/linux-cmd/wildcard/graph4
/home/lar/linux-cmd/wildcard/Graph3
/home/lar/linux-cmd/wildcard/Graph1
/home/lar/linux-cmd/wildcard/graph1
/home/lar/linux-cmd/wildcard/graph3
/home/lar/linux-cmd/wildcard/.my-private-info.txt
/home/lar/linux-cmd/wildcard/graph5
/home/lar/linux-cmd/wildcard/Graph2
```

တကယ်လို့ ရှာခိုင်းချင်တဲ့ ဖိုင်နာမည်ပုံစံ (filename pattern) က တစ်ခုထက်မက ပိုရင် -o (or option) နဲ့ တွဲပြီး ရှာခိုင်းလို့ရပါတယ်။ ဥပမာ ဒေါင်းလုပ်လုပ်ထားတဲ့ ဖိုင်တွေထဲက ".tar.gz" နဲ့ ဆုံးတဲ့ ဖိုင်နာမည် (သို့) "T" စာလုံးနဲ့စပြီး ".zip" နဲ့ ဆုံးတဲ့ဖိုင်တွေကိုပဲ ရာပေးပါ လို့ ခိုင်းချင်ရင် အောက်ပါအတိုင်း command ပေးလို့ရပါတယ်။

In [33]:

```
find ~/Downloads/ -type f \( -name "*.tar.gz" -o -name "T*.zip" \)
```

```
/home/lar/Downloads/The-Nature-of-Code-Examples.zip
/home/lar/Downloads/leptonica-1.74.4.tar.gz
/home/lar/Downloads/The-Nature-of-Code-master.zip
/home/lar/Downloads/Text-Mark-up-master.zip
/home/lar/Downloads/opencv-3.3.0.tar.gz
```

ဖိုင်တွေကို ရာဖွေတဲ့ နေရာမှာ permission option နဲ့ လည်းကန့်သတ်ပြီး ရာဖွေနိုင်ပါတယ်။

In [32]:

```
find -type f -perm 644
```

```
./folderB/otest
./otest.word
./otest
./otest.tag
```

-user option ကိုသုံးပြီး၊ ဖိုင်တွေကို ရာဖွေခိုင်းတဲ့ အခါမှာ user နာမည်နဲ့ (သို့) ဘယ်သူကပိုင်တဲ့ ဖိုင်ကို ရာပေးပါဆိုပြီးလဲ အမိန့်ပေးလို့ရပါ

တယ်။

In [34]:

```
find . -user lar -name "*.txt"
```

```
./all-vi-info.txt
./wildcard/hello.txt
./wildcard/abstract.txt
./wildcard/maik-khe.food.txt
./wildcard/mba.note.txt
./wildcard/acknowledgements.txt
./wildcard/.my-private-info.txt
./wildcard/conclusion.txt
```

find command ရဲ့ -size option ကိုသုံးပြီး၊ ဖိုင်ဆိုက် အရွယ်အစားပြောပြီး ရှာဖွေခိုင်းလို့ရပါတယ်။
အောက်ပါ ဥပမာက လက်ရှိဖိုလ်ဒါအောက်မှာ ရှိတဲ့ ဖိုင်တွေအားလုံးထဲက 300k (300 kilobytes) ထက် ကြီးတဲ့ဖိုင်တွေကို ရှာပြီး ရိုက်ထုတ်ပြခိုင်းတာပါ။

In [18]:

```
find . -type f -size +300k
```

```
./linux-commands.ipynb
./folderB/127-135.pdf
./folderB/otest
./linux-commands.pdf
./linux-commands.tex
./.ipynb_checkpoints/linux-commands-checkpoint.ipynb
./folderA/127-135.pdf
./otest
./linux-commands.log
./linux-commands.ipynb.bak
./all-vi-info.jpg
```

Kilobytes ယူနစ်နဲ့သာမကပဲ bytes ယူနစ်နဲ့ ရှာချင်ရင် "c"၊ Megabytes ယူနစ်နဲ့ ရှာချင်ရင် "M"၊ Gygabytes ယူနစ်နဲ့ ရှာချင်ရင် "G" စတာတွေကို သုံးပြီး ယူနစ်အမျိုးမျိုးနဲ့ ဖိုင်ဆိုက်ကိုပေးပြီး ရှာဖွေလို့ရပါတယ်။ အသေးစိတ်ကို လေ့လာချင်ရင်တော့ ထုံးစံအတိုင်း man find နဲ့ ရှာကြည့်ပါ။

အောက်ပါ ဥပမာက လက်ရှိဖိုလ်ဒါအောက်မှာ ရှိတဲ့ ဖိုင်တွေထဲက ဖိုင်ဆိုက် 2M ထက်ကြီးတဲ့ ဖိုင်ကို ရှာခိုင်းတာပါ။

In [1]:

```
find . -type f -size +2M
```

```
./linux-commands.log
```

အထက်မှာ သုံးပြခဲ့တဲ့ "+300k", "+2M" မှာ "+" သင်္ကေတ ပါနေတာကို သတိပြုမိမယ်လို့ ထင်ပါတယ်။ ဆန့်ကျင်ဘက်အနေနဲ့ "-" (Minux) သင်္ကေတကိုသုံးပြီး ဥပမာ "-2M" လိုပေးခဲ့ရင်၊ ဖိုင်ဆိုက် 2 Megabytes ထက် ငယ်တဲ့ ဖိုင်တွေကို ရှာပေးပါလို့ ဆိုလိုပြောတာပါ။
ဒါ့အပြင်၊ "+" သင်္ကေတရော၊ "-" သင်္ကေတရော နှစ်မျိုးစလုံးကိုသုံးပြီး၊ ဘယ်ဆိုက်ကနေ ဘယ်ဆိုက်ကြားမှာရှိတဲ့ ဖိုင်တွေကို ရှာပေးပါဆိုပြီး လုပ်ခိုင်းလို့ ရပါတယ်။ အောက်ပါ ဥပမာက ဖိုင်ဆိုက် 300k နဲ့ 400k ကြားမှာရှိတဲ့ ဖိုင်နာမည်တွေကို ရိုက်ထုတ်ခိုင်းတာဖြစ်ပါတယ်။

In [19]:

```
find . -type f -size +300k -size -400k
```

```
./folderB/127-135.pdf
./folderB/otest
./linux-commands.tex
./folderA/127-135.pdf
./otest
./linux-commands.ipynb.bak
./all-vi-info.jpg
```

find command ရဲ့နောက်ထပ် အသုံးဝင်တဲ့ option တစ်ခုကတော့ -mtime ပါ။
အောက်ပါ -mtime -2 (minus two) ဥပမာက လွန်ခဲ့တဲ့ ၂ရက်အတွင်းမှာ နောက်ဆုံးပြင်ခဲ့တဲ့ ဖိုင်နာမည်တွေကို ရိုက်ထုတ်ခိုင်းတာပါ။

In [23]:

```
find . -mtime -2
```

```
.
./linux-commands.ipynb
./numbers.txt
./.ipynb_checkpoints/linux-commands-checkpoint.ipynb
```

တကယ်လို့ အထက်ပါ find command မှာ minus သင်္ကေတကို မသုံးပဲ၊ plus သင်္ကေတကိုသုံးပြီး၊ ဥပမာ +50 ဆိုပြီး ပေးခဲ့ရင်တော့၊ လွန်ခဲ့တဲ့ ရက် ၅၀အတွင်း ဘာမှပြင်ဆင်တာမျိုး (modifyလုပ်တာ) မလုပ်ခဲ့တဲ့ ဖိုင်တွေကို ရှာခိုင်းတာပါ။

တခါတလေမှာ ကျွန်တော်အပါအဝင်ပါ။ ကွန်ပြူတာသမားတွေဖြစ်တတ်ကြပါတယ်။
ခုနလေးတင် ဖွင့်ဖတ်ကြည့်လိုက်သေးတယ်။ အဲဒီဖိုင်နာမည်ကို မေ့သွားတယ်။ အဲဒီလိုအခါမျိုးမှာ find command ရဲ့ -amin option က အလွန်ကို အသုံးဝင်ပါတယ်။

အောက်ပါ ဥပမာက လွန်ခဲ့တဲ့ ၅မိနစ်က နောက်ဆုံး access လုပ်ခဲ့တဲ့ .txt နဲ့ဆုံးတဲ့ဖိုင်ကို ရှာခိုင်းတာပါ။

In [44]:

```
find -amin 5 -name "*.txt"
```

```
./fruit.txt
```

နောက်ထပ် အသုံးဝင်တဲ့ option တစ်ခုကတော့ ဖိုင်တစ်ဖိုင်ကိုပေးပြီး အဲဒီဖိုင်ထက် ပိုအသစ်ဖြစ်တဲ့၊ ဆိုလိုတာက ပေးလိုက်တဲ့ဖိုင်ထက် နောက်ကျပြီး ပြင်တာဆင်တာ (modify) လုပ်ထားတဲ့ ဖိုင်တွေကို ရှာခိုင်းတဲ့အခါသုံးတဲ့ -newer ဆိုတဲ့ option ပါ။

အောက်ပါ ဥပမာက fileA ထက် နောက်ကျပြီး modify လုပ်ထားတဲ့ text ဖိုင်တွေကို ရှာခိုင်းတာပါ။

In [38]:

```
find . -newer ./fileA -name "*.txt"
```

```
./empty-file.txt
./myanmar-food.txt
./numbers.txt
./fruit-orange.txt
./random.txt
./fruit.txt
```

-empty ဆိုတဲ့ option ကိုလည်း လေ့လာကြရအောင်။

သူကတော့ သူ့ရဲ့နာမည်အတိုင်းပါပဲ။ empty ဖြစ်တဲ့ ဖိုင်တွေ၊ ဖိုလ်ဒါတွေကို ရာပေးတဲ့ option ပါ။

အောက်ပါ find command ဥပမာကို ကြည့်ပါ။
လက်ရှိ ဖိုလ်ဒါအောက်၊ သူ့ရဲ့ ဆင်ဖိုလ်ဒါအောက်မှာ ရှိနေတဲ့ ဖိုင်အလွတ်တွေ၊ ဖိုလ်ဒါအလွတ်တွေကို ရာခိုင်းတဲ့ ဥပမာပါ။

In [39]:

```
find . -empty
```

```
./folder-new/folder-new
./.fileC
./empty-file.txt
./folder-athit
./data1
./data4
./echo
./data3
./data5
./data2
./wildcard/graph2
./wildcard/Graph4
./wildcard/superman
./wildcard/abstract.txt
./wildcard/graph4
./wildcard/Graph3
./wildcard/maik-khe.food.txt
./wildcard/Graph1
./wildcard/mba.note.txt
./wildcard/matrix
./wildcard/graph1
./wildcard/acknowledgements.txt
./wildcard/m
./wildcard/graph3
./wildcard/.my-private-info.txt
./wildcard/graph5
./wildcard/conclusion.txt
./wildcard/Graph2
./newfile
```

28. grep (Global Regular Expression Print)

grep command က standard input ကနေပေးလိုက်တဲ့ စာကြောင်း (သို့) ဖိုင်တွေထဲမှာ user က ရာခိုင်းတဲ့ စာလုံး၊ စာကြောင်း၊ Regular Expression pattern တွေကို ရာဖွေပေးတဲ့ ပရိုဂရမ်ပါ။ string pattern တွေကို ရာဖွေတဲ့ နေရာမှာ မြန်ဆန်တဲ့ Boyer-Moore algorithm (https://en.wikipedia.org/wiki/Boyer%E2%80%93Moore_string_search_algorithm) ကိုအသုံးပြုထားပါတယ်။

linux system မှာ register လုပ်ထားတဲ့ user အားလုံးရဲ့ user name, encrypted password, user ID number (UID), user's group ID number (GID), full name of the user (GECOS), user home directory, login shell တွေကို သိမ်းထားတဲ့ passwd ဆိုတဲ့နာမည်နဲ့ ဖိုင်ရှိပါတယ်။ အဲဒီဖိုင်ထဲမှာ guest ဆိုတဲ့ user name နဲ့ ပတ်သတ်တဲ့ အချက်အလက်ကို ရာဖွေကြည့်ကြရအောင်။

In [35]:

```
grep guest /etc/passwd
```

guest-a6syh9:x:999:999:Guest:/tmp/**guest**-a6syh9:/bin/bash
guest:x:1001:1001:Guest User,,,:/home/**guest**:/bin/bash

grep command က ရာခိုင်းတဲ့ စာလုံးတွေ၊ RE pattern တွေကို ရာလိုတွေ့ရင် အဲဒီစာလုံးတွေကို အရောင်နဲ့ highlight လုပ်ပြီး ပြပေးပါတယ်။

စာလုံးအကြီး၊ အသေး မခွဲခြားပဲ ရာပေးပါဆိုရင် -i option ကို သုံးပါတယ်။

In [36]:

```
grep -i guest /etc/passwd
```

guest-a6syh9:x:999:999:**Guest**:/tmp/**guest**-a6syh9:/bin/bash
guest:x:1001:1001:**Guest** User,,,:/home/**guest**:/bin/bash

In [1]:

```
grep -w "blood" fileA
```

My **blood** type is O.

In [7]:

```
grep "report" *.txt
```

```
       /usr/share/vim/vim74/bug**report**.vim
                      Script to generate a bug **report**.  See ":help bug
s".
```

-c option နဲ့ ဘယ်နှစ်ခါ ရာတွေ့သလဲ ဆိုတာကို အောက်ပါအတိုင်း ရေတွက်ခိုင်းလို့ရပါတယ်။

In [8]:

```
grep -c "report" *.txt
```

2

-H option နဲ့ဆို ရာခိုင်းခဲ့တဲ့စာလုံး၊ RE pattern တွေကို ရာဖွေလိုတွေ့ရှိတဲ့ဖိုင်ရဲနာမည်ကိုပါ ရိုက်ပြပေးပါလိမ့်မယ်။

In [22]:

```
grep -H "report" *.txt
```

```
all-vi-info.txt:       /usr/share/vim/vim74/bug**report**.vim
all-vi-info.txt:                       Script to generate a bug **report**.
  See ":help bugs".
```

ဘယ်နှစ်ကြောင်းမြောက်မှာ ရာတွေ့တယ် ဆိုတာကို နံပါတ် (line number) နဲ့ ရိုက်ထုတ်ပြခိုင်းချင်ရင် -n option ကို သုံးပါတယ်။

In [67]:

```
grep -in report *.txt
```

```
259:       /usr/share/vim/vim74/bug**report**.vim
260:                       Script to generate a bug **report**.  See ":help
 bugs".
```

grep ကပုံမှန်အားဖြင့် ရာတွေ့တဲ့ စာလုံး၊ ကိုက်ညီတဲ့ RE pattern တွေကို အနီရောင်နဲ့ highlight လုပ်ပြီး ပြပေးပါတယ်။ အရောင်ပြောင်းချင်ရင်တော့ grep command ရဲ့ရှေ့မှာ GREP_COLOR variable တစ်ခုဆောက်ပြီး ကိုယ်ကြိုက်တဲ့ အရောင်နဲ့ ရိုက်ခိုင်းလို့ရပါတယ်။ အရောင်ပြောင်းရိုက်ခိုင်းကြည့်ရအောင်။

In [66]:

```
GREP_COLOR='01;35' grep -in report *.txt
```

```
259:       /usr/share/vim/vim74/bug**report**.vim
260:                       Script to generate a bug **report**.  See ":help
 bugs".
```

-B option နဲ့ ရာလို့တွေတဲ့ စာကြောင်းရဲ့အထက်မှာရှိတဲ့ ဘယ်နှစ်ကြောင်းကိုလည်း တွဲပြီးရိုက်ပြပေးပါ ဆိုပြီး ခိုင်းလို့ရပါတယ်။ တချို့အလုပ်တွေမှာက ရာဖွေလို့တွေ့တဲ့ စာကြောင်းတစ်ကြောင်းထဲကို ပြပေးတာထက် သူ့အထက်က စာကြောင်း၊ သူ့အောက်က စာကြောင်းတွေနဲ့ပါတွဲပြီး ကြည့်မှ အဆင်ပြေတဲ့ အခါမျိုး ရှိတတ်လို့ပါ။

အောက်ပါ ဥပမာက -B 2 option နဲ့ ရာတွေ့တဲ့ စာကြောင်းရဲ့အထက် နှစ်ကြောင်းကိုပါ အတူတူရိုက်ပြပေးပါလို့ option ပေးတာပါ။

In [25]:

```
grep -B 2 "blood" fileA
```

```
I am fileA.
I was born in April.
My **blood** type is O.
```

-A option နဲ့ grep ပရိဂရမ်ကို ရာခိုင်းလို့သွားတွေ့တဲ့ စာကြောင်းရဲ့အောက်ဘက် ဘယ်နှစ်ကြောင်းရိုက်ပြပေးပါဆိုပြီး လဲ လုပ်ခိုင်းလို့ရပါတယ်။

အောက်ပါ ဥပမာက -A 3 option နဲ့ သွားတွေ့တဲ့ စာကြောင်းသာမက သူ့ရဲ့အောက်ဖက်က စာကြောင်း သုံးကြောင်းကိုပါ တွဲရိုက်ပြပေးပါဆိုပြီး အမိန့်ပေးတာပါ။

In [27]:

```
grep -A 3 "Hi" fileB
```

```
**Hi**!
I am fileB.
I was born in November.
My blood type is O.
```

တခြား linux command တွေလိုပါပဲ။ grep မှာလည်း option တွေကို တခုထက်မက တွဲပြီး အလုပ်လုပ်ခိုင်းလို့ရပါတယ်။ -B နဲ့ -A ကိုတွဲပြီး run ကြည့်ရအောင်။

In [30]:

```
grep -B 1 -A 1 "November" fileB
```

```
I am fileB.
I was born in November.
My blood type is O.
```

-B နဲ့ -A option နှစ်ခုအစား -C option ကိုလည်း သုံးရင်လည်း အလုပ် အတူတူလုပ်ပေးပါလိမ့်မယ်။ အောက်ပါ ဥပမာကို ကြည့်ပါ။

In [36]:

```
grep -C 1 "November" fileB
```

```
I am fileB.
I was born in November.
My blood type is O.
```

-o option နဲ့ ရာခိုင်းလို့ သွားတွေ့တဲ့စာကြောင်း တစ်ကြောင်းလုံးကို ရိုက်မပြစေချင်ပဲ၊ တွေ့တဲ့စာလုံးကိုပဲ သီးသန့်ရိုက်ထုတ်ပါ ဆိုပြီးလဲ ခိုင်းလို့ရပါတယ်။ အောက်ပါ ဥပမာက လက်ရှိဖိုလ်ဒါအောက်မှာ ရှိတဲ့ .txt နဲ့ ဆုံးတဲ့ ဖိုင်တွေထဲမှာ "This" ဆိုတဲ့ စာလုံးကို ဘယ်နှစ်ခါသုံးထားသလဲဆိုတာကို သိချင်တဲ့ အခါမျိုးမှာ အသုံးပြုပါတယ်။

In [33]:

```
grep -o "This" *.txt
```

```
This
This
This
This
This
This
This
This
This
```

ကျွန်တော်တို့ အရေအတွက် ဘယ်လောက်ရှိသလဲဆိုတာကို အထက်မှာ သင်ပေးခဲ့တဲ့ wc command ကို pipe နဲ့ ပို့ပေးပြီး ရေတွက်ခိုင်းလို့ ရတယ်လေ။

In [34]:

```
grep -o "This" *.txt | wc
```

```
      9       9      45
```

29. egrep (Extended grep)

egrep က grep command ကို -E option နဲ့တွဲသုံးတာနဲ့ အတူတူပါပဲ။ ပေးလိုက်တဲ့ pattern ကို Extended Regular Expression Set အနေနဲ့ ဖတ်ပြီးအလုပ်လုပ်ပေးပါတယ်။

ဥပမာ grep command ကို အောက်ပါအတိုင်း blood စာလုံး (သို့) Hi ဆိုတဲ့ စာလုံးကို ရှာပေးပါလို့ ခိုင်းရင် အလုပ်မလုပ်ပေးတာကို တွေ့ရပါလိမ့်မယ်။ ဘာ့ကြောင့်လဲ ဆိုတော့ blood|Hi ကို စာလုံးတစ်လုံးလို ဖတ်ပြီး ရှာပေးလို့ပါ။

In [41]:

```
grep -w 'blood|Hi' fileA fileB
```

egrep command ကတော့ "|" ကို "or" လို နားလည်ပေးပါတယ်။ RE အနေနဲ့ ဖတ်ပြီး အလုပ်လုပ်ပေးပါတယ်။

In [42]:

```
egrep -w 'blood|Hi' fileA fileB
```

```
fileA:My blood type is O.
fileB:Hi!
fileB:My blood type is O.
```

အထက်ပါ အလုပ်ကို အောက်ပါအတိုင်း grep command ကို -E နဲ့ option နဲ့ တွဲပြီး လုပ်ခိုင်းလို့ရပါတယ်။ ရလဒ်က အတူတူပဲ ဖြစ်ပါလိမ့်မယ်။

In [39]:

```
grep -E -w 'blood|Hi' fileA fileB
```

```
fileA:My blood type is O.
fileB:Hi!
fileB:My blood type is O.
```

ဘာဖြစ်လို့ ဒီလို grep -E နဲ့ လုပ်လို့ရရဲ့သားနဲ့ egrep ဆိုပြီး ပရိဂရမ်သပ်သပ်ရိုနေရတာလဲဆိုရင် ဟိုးအရင်က grep နဲ့ egrep ဆိုပြီး ပရိဂရမ်သပ်သပ်စီ ရိုနေခဲ့ကြလို့ပါ။ နောက်ပိုင်းကြမှ grep မှာ egrep လိုမျိုးအလုပ်လုပ်လို့ရအောင် -E option ဆိုပြီး ထပ်ထည့်တာမို့လို့ပါ။

RE pattern တွေနဲ့ ရှာတာဖွေတာလုပ်မယ် ဆိုရင် egrep (သို့) grep -E နှစ်မျိုးစလုံးနဲ့ အသုံးပြုနိုင်ပါတယ်။

လက်တွေ့ RE pattern တစ်ခုဖြစ်တဲ့ "rep.*t" ကို သုံးကြည့်ရအောင်။

In [5]:

```
egrep 'rep.*t' *.txt
```

```
       /usr/share/vim/vim74/bug**report**.vim
                     Script to generate a bug **report**.  See ":help bug
s".
       Note  that  a  number  of things that may be regarded as bugs b
y some, are in fact caused by a too-faithful **reproduction of Vi's beha
**viour.  And if you think other t**hings are bugs "because Vi
```

30. sed (stream editor for filtering and transforming text)

ဖိုင်ထဲမှာရှိတဲ့ စာသားတွေကို အပြောင်းအလဲလုပ်ဖို့အတွက်ဆိုရင် sed command က အသုံးဝင်ပါတယ်။ တကယ့်ကို powerful ဖြစ်တဲ့ tool တစ်ခုလည်းဖြစ်ပါတယ်။ သုံးပုံသုံးနည်း တချို့ကို ဒီနေရာမှာ မိတ်ဆက်ပါမယ်။

sed program မှာ command တွေအများကြီးရှိပါတယ်။ ကျွန်တော်အသုံးများတာကတော့ ဖိုင်ထဲမှာ ရှိနေတဲ့ စာလုံးတွေကို တခြားစာလုံးနဲ့ အစားထိုးတဲ့အခါမှာသုံးတဲ့ substitute command ပါ။ အတိုကောက် အနေနဲ့ "s" ပါ။ RE ကို အသုံးပြုသလို slash သုံးခုပုံစံ (s/ရှိနေတဲ့ စာလုံး/အစားထိုးမဲ့စာလုံး/) ပါပဲ။

ဥပမာ fruit.txt ဖိုင်ထဲမှာရှိနေတာက mango ဆိုပါစို့။ mango ကို orange နဲ့ အစားထိုးမယ်ဆိုရင် အောက်ပါအတိုင်း sed command ပေးခိုင်းလို့ရပါတယ်။

အရင်ဆုံး fruit.txt ဖိုင်ထဲမှာ ဘာတွေရှိသလဲဆိုတာကို cat နဲ့ ကြည့်ရအောင်

In [1]:

```
cat ./fruit.txt
```

```
I love mango.
I ate two mango.
mango is mango.
```

s/mango/orange/ နဲ့ fruit.txt ဖိုင် ထဲက mango တွေကို orange နဲ့အစားထိုးပြီး fruit-orange.txt ဖိုင်ထဲမှာ သိမ်းပါမယ်။

In [3]:

```
sed s/mango/orange/ fruit.txt > fruit-orange.txt
```

fruit-orange.txt ဖိုင်ထဲမှာ ဘယ်လိုပြောင်းလဲသွားသလဲ ဆိုတာကို ကြည့်ရအောင်။

In [5]:

```
cat fruit-orange.txt
```

```
I love orange.
I ate two orange.
orange is mango.
```

အထက်ပါ output ကိုကြည့်ပြီးနောက်ဆုံးစာကြောင်းက ဒုတိယmango ကို orange အဖြစ်ပြောင်းမပေးတာကို သတိထားမိမယ်လို့ထင်ပါတယ်။ အဲဒါက substitute လုပ်ခိုင်းတဲ့ အခါမှာ g (global) ဆိုတဲ့ parameter ကိုမပေးခဲ့လို့ပါ။ တကယ်လို့ ကျွန်တော်တို့က g parameter ကိုသာပေးခဲ့မယ်ဆိုရင် စာကြောင်းတစ်ကြောင်းမှာ ပထမဆုံးတွေ့တဲ့ mango ကို orange နဲ့ အစားထိုးပြီး ရပ်မသွားပဲ၊ နောက်ထပ် ထပ်တွေ့တဲ့ ဒုတိယ mango ကိုလည်း orange နဲ့ အစားထိုးပေးပါလိမ့်မယ်။ s/orange/mango/g နဲ့ run ကြည့်ရအောင်။

In [8]:

```
sed s/mango/orange/g fruit.txt > fruit-orange.txt
```

In [9]:

```
cat ./fruit-orange.txt
```

```
I love orange.
I ate two orange.
orange is orange.
```

အခုဆိုရင် mango အားလုံးကို orange အဖြစ် ပြောင်းပေးသွားတာတွေ့ရပါမယ်။

ပုံမှန်အားဖြင့် စာလုံးတစ်လုံးကို အခြားစာလုံးတစ်လုံးနဲ့ အစားထိုးတဲ့အခါမှာ s ကိုရိုက်ပြီး slash (/) သုံးခုနဲ့ ရေးတဲ့ syntax ဖြစ်တဲ့ 's/old_word/new_word/' ဆိုတဲ့ ပုံစံကို သုံးကြပေမဲ့၊ နောက်ထပ် အသုံးဝင်တဲ့ ပုံစံတခုကို ရှင်းပြပါမယ်။

တကယ်တမ်းက ကျွန်တော်တို့ slash အစား တခြားစာလုံးကိုလည်း delimiter အဖြစ်သုံးလို့ရပါတယ်။ တစ်ခါတလေ အဲဒါကပိုပြီးတော့ အဆင်ပြေပါတယ်။ ဥပမာ website URL တွေလို အစားထိုးမဲ့ စာကြောင်းမှာ slash တွေပါနေတဲ့ အခါမျိုးမှာပါ။

အောက်ပါ ဥပမာက ".com/burmese" ကို ".org/myanmar" နဲ့ အစားထိုးတဲ့ ဥပမာပါ။

In [8]:

```
echo "http://www.bbc.com/burmese" | sed 's_com/burmese_org/myanmar_'
```

```
http://www.bbc.org/myanmar (http://www.bbc.org/myanmar)
```

Excution Cycle of sed

sed နဲ့ input stream ထဲမှာရှိတဲ့ စာသားတွေထဲက၊ ကိုယ်လိုချင်တဲ့ အစိတ်ပိုင်းတစ်ပိုင်းကိုပဲ ရိုက်ထုတ်တာ၊ စာလုံးတွေကို ဝင်ပြင်တာ၊ format ပြောင်းတာ စသည်ဖြင့် လုပ်လို့ရတဲ့ အလုပ်တွေအများကြီးရှိပါတယ်။ အဲဒီလို အလုပ်တွေအများကြီးကို command line ကနေ (သို့) bash shell script တွေထဲကနေ သုံးဖို့ဆိုရင်၊ sed command ရဲ့အလုပ်လုပ်တဲ့ပုံစံ (excution cycle) ကို သိထားသင့်ပါတယ်။

sed မှာ ဝင်လာတဲ့ဒေတာတွေကို ကိုင်တွယ်ပြီး အလုပ်လုပ်ဖို့အတွက် pattern space နဲ့ hold space ဆိုပြီး data buffer နှစ်မျိုးရှိပါတယ်။ Pattern space က temporary buffer ပါပဲ။ လက်ရှိလုပ်နေတဲ့ အလုပ်နဲ့ ပတ်သက်တဲ့ ဒေတာ (သို့) စာကြောင်းကို သိမ်းထားပါတယ်။ Hold space ကတော့ လိုတဲ့အခါပြန်ယူပြီးသုံးဖို့အတွက် သိမ်းထားတဲ့ long term buffer ဖြစ်ပါတယ်။ sed က နောက်လာမဲ့ command တွေမှာ အရင်လုပ်ခဲ့တဲ့ ဒေတာနဲ့ ပတ်သက်ပြီး နှိုင်းယှဉ်ကြည့်ဖို့လိုအပ်တဲ့ အခါ၊ အရင်ဒေတာကိုပါယူသုံးရတဲ့ အခါမျိုးမှာ hold space ကနေ ယူသုံးပါတယ်။

အလုပ်လုပ်တဲ့အခါမှာ ကျွန်တော်တို့က Hold space ကို တိုက်ရိုက်ကိုင်တွယ်ပြီး အလုပ်မလုပ်ပါဘူး။ Hold space ကဒေတာကို လိုအပ်တဲ့အခါမှာ pattern space ပေါ်ကို ကော်ပီကူး ပြီးသုံးတာမျိုး၊ pattern space မှာရှိနေတဲ့ ဒေတာနဲ့ ပေါင်းစပ်ပြီး (append to pattern space) သုံးတာမျိုးပဲ လုပ်ကြပါတယ်။

sed ပရိုဂရမ်ကို စာကြောင်းတွေ၊ စာပိုဒ်တွေကို ရိုက်ထုတ်ခိုင်းတဲ့ အခါမျိုး မှာ pattern space က ဒေတာကို ရိုက်ထုတ်ပေးတာပါ။ အထက်မှာ sed အသုံးပြုပုံ ဥပမာအဖြစ် ပြခဲ့တဲ့ substitute အလုပ်ကလည်း pattern space ပေါ်မှာပဲ အလုပ်လုပ်သွားတာပါ။ မော်နီတာမှာ "p" နဲ့ ရိုက်ထုတ်ခိုင်းတဲ့ အခါမှာလဲ pattern space က ဒေတာကိုပဲ ရိုက်ထုတ်ပေးတာပါ။

sed က pattern space, hold space ကို သုံးပြီး အလုပ်လုပ်တဲ့ ပုံစံကို နားလည်လွယ်အောင် လက်တွေ့ဥပမာနဲ့ ရှင်းပြပါမယ်။

စာကြောင်းသုံးကြောင်း (line 1., line 2. နဲ့ line 3.) ကို echo command နဲ့ ရိုက်ထုတ်ကြည့်ရင် အောက်ပါအတိုင်း မြင်ရပါလိမ့်မယ်။

In [8]:

```
echo -e "line 1.\nline 2.\nline 3."
```

```
line 1.
line 2.
line 3.
```

ဆိုကြပါစို့ အထက်ပါစာကြောင်းသုံးကြောင်းကို input stream တစ်ခုအနေနဲ့ sed ပရိုဂရမ်ဆီကို "|" (piping) နဲ့ ပို့ပေးပြီး -n '1!G;h;$p' ဆိုတဲ့ command ပေးရင် အောက်ပါအတိုင်း output ကို မြင်ရပါလိမ့်မယ်။

In [7]:

```
echo -e "line 1.\nline 2.\nline 3."| sed -n '1!G;h;$p'
```

```
line 3.
line 2.
line 1.
```

အထက်ပါအတိုင်း echo နဲ့ pass လုပ်ပေးလိုက်တဲ့ စာကြောင်းသုံးကြောင်းကို sed က ပြောင်းပြန် ရိုက်ထုတ်ပေးတာကို မြင်ရပါလိမ့်မယ်။

'1!G;h;$p' ဆိုတဲ့ စာကြောင်းမှာ 1!G ရယ် h နဲ့ $p ဆိုပြီး command သုံးခုပါဝင်ပါတယ်။

1!G; မှာ "G" က hold space ထဲက ဒေတာကို pattern space ဆီကို ပေါင်းထည့်ပေး (append) လုပ်ပေးပါတယ်။ "1" က ပထမဆုံး စာကြောင်း ကို ညွှန်းတာပါ။ အလုပ်လုပ်မဲ့ စာကြောင်းကို ညွှန်းတာ ပါ။ address ပေးတာလို့လည်း ပြောကြပါတယ်။ "!" သင်္ကေတက ပေးလိုက်တဲ့ အမိန့်ကို ဆန့်ကျင်ဘက်၊ not (သို့) negate လုပ်ပေးပါလို့ ခိုင်းတာပါ။ ဒီနေရာမှာတော့ "1!G" မို့လို့ ပထမဆုံးစာကြောင်းကလွဲ

ပြီး ကျန်တဲ့စာကြောင်းတိုင်းကို pattern space ဆီကို append လုပ်ပေးပါလို့ ဆိုလိုပါတယ်။

နောက်ထပ် အမိန့်တခုဖြစ်တဲ့ "h" ကတော့ pattern space မှာရှိနေတဲ့ ဒေတာတွေကို hold space ဆီကို ကော်ပီကူးပေးပါလို့ ဆိုလိုပါတယ်။ အဲဒီလိုလုပ်တဲ့အခါမှာ hold space ထဲမှာရှိတဲ့ ဒေတာတွေကို overwrite လုပ်သွားပါလိမ့်မယ်။ ဒီ "h" command ကတော့ pattern space စာကြောင်းတွေအားလုံးကို လုပ်ပေးပါလိမ့်မယ်။

$p မှာ "$"ကတော့ နောက်ဆုံးစာကြောင်းကို ညွှန်းပါတယ်။ "p" ကတော့ လက်ရှိ pattern space ကို မော်နီတာစကရင်မှာ ရိုက်ထုတ်ပေးပါလို့ခိုင်းတာပါ။ အဲဒါကြောင့် "$p" က နောက်ဆုံးစာကြောင်းကို ရောက်ရှိတဲ့ အချိန်မှာ pattern space ကို print လုပ်ပေးပါလို့ ဆိုလိုပါတယ်။

အထက်ပါ '1!G;h;$p' ရဲ့အလုပ်လုပ်ပုံကို ပိုပြီးတော့ မြင်သာအောင် sed command က တစ်ကြောင်းပြီး တစ်ကြောင်း ဖတ်ပြီး command တွေကို run နေစဉ်မှာ pattern space/hold space ထဲမှာ ရှိနေမဲ့ ဒေတာကို ဇယားပုံစံမျိုးနဲ့ ပြသပါမယ်။

ဝင်လာတဲ့ စာကြောင်း	Pattern space နဲ့ Hold space ထဲကဒေတာ	Run ခဲ့တဲ့ command
line1	line1.$	h
line2	line2.\nline1.$	1!G;h
line3	line3.\nline2.\nline1.$	1!G;h;$p

တကယ်လို့ pattern space ရဲ့အပြောင်းအလဲကို တစ်ကြောင်းပြီး တစ်ကြောင်းကြည့်ချင်ရင် "l" command ကိုသုံးပြီး ကြည့်နိုင်ပါတယ်။

In [20]:

```
echo -e "line 1.\nline 2.\nline 3." | sed -n '1!G;h;l;$p'
```

```
line 1.$
line 2.\nline 1.$
line 3.\nline 2.\nline 1.$
line 3.
line 2.
line 1.
```

Some more examples of using sed

"q" command က လက်ရှိလုပ်နေတဲ့ sed ရဲ့အလုပ်ကို ရပ်ခိုင်းတဲ့ "quit" (သို့) "exit-code" ဖြစ်ပါတယ်။ "2q" ဆိုပြီး command ပေးရင် pattern space ထဲမှာ ရှိတဲ့ စာကြောင်းတွေထဲက နှစ်ကြောင်းအထိ မော်နီတာစကရင်မှာ ရိုက်ပေးပြီးရင် quit လုပ်သွားပါလိမ့်မယ်။

In [9]:

```
echo -e "line 1.\nline 2.\nline 3."| sed 2q
```

```
line 1.
line 2.
```

"d" command က pattern space ကိုဖျက်ပေးပါတယ်။ ဥပမာ "1d" ဆိုပြီး command ပေးရင် pattern space ထဲမှာရှိတဲ့စာကြောင်းတွေထဲက ပထမဆုံးစာကြောင်းကို ဖျက်ပြီး ကျန်တာကို မော်နီတာစကရင်မှာ ရိုက်ထုတ်ပြပေးပါလိမ့်မယ်။

In [10]:

```
echo -e "line 1.\nline 2.\nline 3."| sed 1d
```

```
line 2.
line 3.
```

sed ရဲ့ "p" command နဲ့ ရိုက်ချင်တဲ့ စာကြောင်းကို ရိုက်ခိုင်းလို့ရပါတယ်။ "3p" က pattern space ထဲက စာကြောင်း သုံးကြောင်း မြောက်ကိုရိုက်ခိုင်းတာပါ။ အဲဒီလိုရိုက်ခိုင်းတဲ့ အခါမှာ ပုံမှန်အားဖြင့် pattern space ထဲမှာရှိနေတဲ့ စာကြောင်း၊ စာပိုဒ် စတာတွေ အားလုံးကိုပါ မော်နီတာ စကရင်မှာရိုက်ပေးပါတယ်။ အဲဒီလို မရိုက်အောင် -n (သို့) --quiet (သို့) --silent နဲ့ တားဆီးနိုင်ပါတယ်။ "p" command က "-n" option နဲ့ အမြဲတွဲသုံးပါတယ်။

In [19]:

```
echo -e "line 1.\nline 2.\nline 3."| sed -n 3p
```

```
line 3.
```

"-n" option ကို ဖြုတ်ပြီး run ကြည့်ရင် အောက်ပါအတိုင်း pattern space ပေါ် မှာရှိနေတဲ့ စာကြောင်းသုံးကြောင်းကိုပါ အတူတူတွဲရိုက်ပေးတာကို တွေ့ရပါလိမ့်မယ်။

In [15]:

```
echo -e "line 1.\nline 2.\nline 3."| sed 3p
```

```
line 1.
line 2.
line 3.
line 3.
```

command တစ်ခုထက်မက ကိုပေးချင်တယ်ဆိုရင်၊ command တွေကို တွန့်ကွင်း နှစ်ခုအကြားမှာ ထည့်ပြီးပေးရပါတယ်။ bash shell script တွေမှာ သုံးတဲ့ ပုံစံအတိုင်း၊ command တစ်ခုနဲ့ တစ်ခုအကြားမှာတော့ ";" နဲ့ခြားပေးရပါတယ်။ အောက်ပါ ဥပမာက "1p" နဲ့ "3p" command နှစ်ခုနဲ့ စာကြောင်းနံပါတ် ၁ နဲ့ စာကြောင်းနံပါတ် ၃ ကို ရိုက်ခိုင်းတာပါ။

In [18]:

```
echo -e "line 1.\nline 2.\nline 3."| sed -n '{1p;3p}'
```

```
line 1.
line 3.
```

အထက်မှာ သင်ကြားခဲ့သလို head နဲ့ tail command နှစ်ခုကို သုံးပြီး၊ ဖိုင်ထဲမှာ ရှိတဲ့စာကြောင်းတွေထဲက ဘယ်စာကြောင်းကနေ ဘယ်စာကြောင်းအထိ ရိုက်ပေးပါလို့ ခိုင်းတဲ့ကိစ္စမျိုးကို sed command တစ်ခုထဲနဲ့ အောက်ပါအတိုင်း ခိုင်းလို့ရပါတယ်။

In [4]:

```
 sed -n '201,205p' otest.tag
```

```
n adj n part v part part punc
n n ppm pron ppm n v ppm punc
v part part pron ppm v part punc
n n n n ppm n v n n ppm adj v n part v conj n adv v ppm punc
conj n n ppm n conj v part v conj v ppm punc
```

အထက်ပါ '201,205p' က otest.tag ဖိုင်ထဲကနေ စာကြောင်း နံပါတ် ၂၀၁ ကနေ ၂၀၅ အထိ ရိုက်ထုတ်ပေးပါလို့ ခိုင်းတာ ဖြစ်ပါတယ်။

Using RE with sed command

လက်တွေ့မှာ Regular Expression နဲ့ sed ကို ကောင်းကောင်း တွဲသုံးကြပါတယ်။
syntax ကတော့ လွယ်လွယ်ရှင်းပါဆိုရင် /RE_pattern/action ဆိုတဲ့ ပုံစံပါ။ ဒီနေရာမှာ action ဆိုတာက sed command က support လုပ်ထားတဲ့ "p" (print), "d" (delete) စတဲ့ command တွေပါ။

ရှေ့မှာ grep command နဲ့ လုပ်ခဲ့သလိုပဲ၊ RE နဲ့ ကိုယ်ရှာချင်တဲ့ စာလုံးရှိတဲ့ စာကြောင်းကိုပဲ sed command ကို ရိုက်ထုတ်ခိုင်းချင်တယ် ဆိုရင်

In [81]:

```
sed -n  '/Mingalar/p' ./fileA
```

```
Mingalar bar!
```

blood ဆိုတဲ့ စာလုံးပါတဲ့ စာကြောင်းကိုတွေ့ရင်၊ အဲဒီလိုင်းနံပါတ်ကို ရိုက်ထုတ်ပြပါ။

In [121]:

```
sed -n '/blood/=' ./fileA
```

```
4
```

ကိုယ်ဖျက်စေချင်တဲ့ စာလုံးကို RE ရဲ့ substitute လုပ်တဲ့ pattern ကို သုံးပြီး လဲ အောက်ပါအတိုင်း ဖျက်ခိုင်းလို့ ရပါတယ်။

In [83]:

```
sed 's/Mingalar//g' ./fileA
```

```
 bar!
I am fileA.
I was born in April.
My blood type is O.
```

အထက်ပါ 's/Mingalar//g' REက Mingalar ဆိုတာကို စာလုံးတစ်လုံး "word" အနေနဲ့ ကြည့်တာ မဟုတ်ပါဘူး။ အဲဒါကြောင့် တကယ်လို့ စာကြောင်းက Mingalarbar လိုဖြစ်နေရင်လည်း ဖျက်မှာပါ။ Mingalar ဆိုတဲ့ စာလုံးကိုပဲ ဖျက်ပေးစေချင်ရင်တော့ RE ရဲ့ word boundary သင်္ကေတဖြစ်တဲ့ less than (<) နဲ့ greater than (>) နှစ်ခုအကြားကို ဖျက်စေချင်တဲ့ စာလုံးကိုထည့်ပြီးရေးရပါမယ်။ အောက်ပါ ဥပမာကို လေ့လာပါ။

In [86]:

```
sed -e 's/\<Mingalar\>//g' ./fileA
```

```
 bar!
I am fileA.
I was born in April.
My blood type is O.
```

တကယ်လို့ Mingalar ဆိုတဲ့ စာလုံးပါတဲ့ စာကြောင်းတစ်ကြောင်းလုံးကို ဖျက်ပေးစေချင်ရင်တော့ d (delete) command ကို သုံးနိုင်ပါတယ်။

In [84]:

```
sed '/Mingalar/d' ./fileA
```

```
I am fileA.
I was born in April.
My blood type is O.
```

နောက်ထပ် RE patten ကိုသုံးတဲ့ဥပမာ တစ်ခုအနေနဲ့ "b" ပါတဲ့ စာကြောင်းကို တွေ့ရင်၊ အဲဒီစာကြောင်းရဲ့အထက်မှာ blank line တစ်ကြောင်းထည့်ပေးစေချင်ရင် အောက်ပါအတိုင်း command ပေးလို့ ရပါတယ်။

In [23]:

```
echo -e "a\nb\nc\nd\n" | sed '/b/{x;p;x;}'
```

```
a

b
c
d
```

အထက်ပါ sed command ဖြစ်တဲ့ "{x;p;x;}" မှာ "x" command က hold space နဲ့ pattern space ကို exchange လုပ်ပေးပါတယ်။ "b" စာကြောင်း ကိုတွေ့တဲ့ အချိန်မှာ၊ hold space ထဲမှာက ဘာမှ မရှိတဲ့ အနေအထားပါ။ လက်ရှိ pattern space မှာက b စာကြောင်းက ရှိနေပါတယ်။ exchange လုပ်လိုက်တဲ့ အခါမှာတော့ pattern space မှာ ဘာမှ မရှိတော့ပဲ၊ hold space မှာ "b" ရှိပါလိမ့်မယ်။ အဲဒီလိုအခြေအနေမှာ "p" command နဲ့ pattern space ကို print လုပ်လို့ blank line ကို မော်နီတာမှာ ရိုက်ထုတ်ပေးပါလိမ့်မယ်။ ပြီးတော့ "x" command နဲ့ hold space နဲ့ pattern space ကို ဒုတိယအခေါက် exchange လုပ်ပါတယ်။ အဲဒါကြောင့် pattern space ဆီကို "b" ပြန်ရောက်သွားပါတယ်။

နောက် ဥပမာတစ်ခုကို ကြည့်ရအောင်။ ဒီတစ်ခါတော့ "G" command ကို သုံးကြည့်ပါမယ်။

In [24]:

```
echo -e "a\nb\nc\nd\n" | sed '/b/G'
```

```
a
b

c
d
```

အထက်ပါ output အတိုင်း RE pattern ဖြစ်တဲ့ b ကို တွေ့တဲ့အခါ blank line တစ်ကြောင်းရိုက်ပေးပါလိမ့်မယ်။

"G" command ရဲ့အလုပ်လုပ်ပုံကတော့ အရင်ဆုံး \n (new line) ကို pattern space မှာ ပေါင်းထည့်ပေး (append) ပါတယ်။ ပြီးတော့ hold space မှာ ရှိတဲ့ ဒေတာကို pattern space မှာ ပေါင်းထည့်ပေးပါတယ်။ အဲဒီအချိန်မှာ hold space မှာက ဘာမှရှိမနေတော့ blank line ကို append လုပ်သွားပါလိမ့်မယ်။

အထက်မှာ ဥပမာအဖြစ် run ပြခဲ့တဲ့ sed '/b/{x;p;x;}' နဲ့ sed '/b/G' နှစ်ကြောင်းကို ပေါင်းပြီး sed '/b/{x;p;x;G;}' အနေနဲ့ run ကြည့်ပါ။

In [25]:

```
echo -e "a\nb\nc\nd\n" | sed '/b/{x;p;x;G;}'
```

```
a

b

c
d
```

အထက်ပါ output အတိုင်း "b" စာကြောင်းကို တွေ့တဲ့အခါမှာ အပေါ် နဲ့အောက်မှာ blank line တစ်ကြောင်းစီ ရိုက်ပေးပါလိမ့်မယ်။

Address and address range of sed

sed command က ဘာမှာ မပြောရင် ဖိုင် (သို့) input လုပ်ပေးတဲ့ stream ထဲမှာရှိတဲ့ စာကြောင်းတိုင်းကို တစ်ကြောင်းချင်းစီဖတ်ပြီး ခိုင်းတဲ့ command အတိုင်း အလုပ်လုပ်ပေးပါတယ်။

တကယ်လို့ ကျွန်တော်တို့က ဘယ်စာကြောင်းကိုပဲ၊ ဘယ်စာကြောင်းကနေ ဘယ်စာကြောင်းအထိ စသည်ဖြင့် သတ်မှတ်ပြီး ပေးလိုက်တဲ့ command ကို apply လုပ်ခိုင်းလို့ရပါတယ်။ address (သို့) address range ပေးတယ်လို့ ပြောပါတယ်။

ကိုယ်က အလုပ်လုပ်စေချင်တဲ့ စာကြောင်းတွေကို နံပါတ်နဲ့ ညွှန်းလို့ရပါတယ်။ address ပေးလို့ရပါတယ်။
အောက်ပါ sed command ရဲ့ '1d;3d;5d;' က စာကြောင်းနံပါတ် ၁၊ ၃ နှင့် ၅ ကို ဖျက်ပေးပါလို့ခိုင်းတာပါ။ အဲဒါကြောင့် brace expansion {1..6} က အမှန်တကယ် line1 ကနေ line6 အထိ စုစုပေါင်း စာကြောင်းရေ ခြောက်ကြောင်း generate လုပ်ပေးပေမဲ့၊ line2, line4 နဲ့ line6 ကိုပဲ ရိုက်ထုတ်ပြတာပါ။ '1d;3d;5d;' ရဲ့ "1", "3" နဲ့ "5" က address ပါ။

In [51]:

```
echo -e line{1..6}'\n'| sed '1d;3d;5d;'
```

```
line2
line4
line6
```

address range ပေးတဲ့ ကိစ္စကို လုပ်ကြည့်ရပေးအောင်။
စာကြောင်းနံပါတ် ၁ ကနေ ၃ အထိဖျက်ပေးစေချင်ရင် '1,3d' ဆိုပြီး ကော်မာနဲ့ခြားပြီး ညွှန်းလို့ရပါတယ်။

In [53]:

```
echo -e line{1..6}'\n'| sed '1,3d'
```

```
line4
line5
line6
```

အောက်ပါ sed ဥပမာက စာကြောင်းနံပါတ်၁ကို ဖျက်ပေးပြီး၊ နောက်ထပ် သုံးကြောင်း ဆက်ပြီးဖျက်ပေးပါလို့ ခိုင်းခဲ့ပါတာ။

In [54]:

```
echo -e line{1..6}'\n'| sed '1,+3d'
```

```
line5
line6
```

အောက်ပါ '2,5!d' ဆိုတာက စာကြောင်းနံပါတ် ၂ ကနေ ၅ ကလွဲရင် ကျန်တဲ့ စာကြောင်းတွေကို ဖျက်ပေးပါလို့ ခိုင်းတာပါ။ အဲဒါကြောင့် စာကြောင်းရေ နံပါတ် ၁ နဲ့ ၆ ကို ဖျက်ပေးပါလိမ့်မယ်။

In [55]:

```
echo -e line{1..6}'\n'| sed '2,5!d'
```

```
line2
line3
line4
line5
```

အောက်ပါ '1~4d' command က စာကြောင်းနံပါတ် ၁ကိုဖျက်ပြီးရင် စာကြောင်းနံပါတ်၄ အထိခုန်ပြီး၊ စာကြောင်းနံပါတ်၅ ကိုဖျက်ပေးပါလိမ့်မယ်။

In [57]:

```
echo -e line{1..6}'\n'| sed '1~4d'
```

```
 line2
 line3
 line4
 line6
```

အောက်ပါ sed '4,2d' နဲ့ ဆိုရင် စာကြောင်းရေနံပါတ်၄ ကိုပဲ ဖျက်ပေးပြီး၊ စာကြောင်းရေ နံပါတ် ၂ ကို ဖျက်မပေးတာတွေ့ရပါလိမ့်မယ်။ ဘာကြောင့်လဲဆိုတော့ sed command က reverse direction အလုပ်မလုပ်နိုင်လို့ပါ။

In [2]:

```
echo -e line{1..6}'\n'| sed '4,2d'
```

```
line1
 line2
 line3
 line5
 line6
```

sed မှာလည်း character class တွေရှိပါတယ်။ ဥပမာ ဝင်လာမဲ့ စာကြောင်းတွေထဲက နံပါတ်နဲ့ဆုံးတဲ့ စာကြောင်းတွေကိုပဲ ရိုက်ပြပေးစေချင်ရင် sed -n '/[[:digit:]]$/p' ဆိုပြီး command ပေးလို့ရပါတယ်။ ဒီနေရာမှာ [[:digit:]] ဆိုတာ နံပါတ် [0-9] ကိုဆိုလိုပြီး၊ ရှေ့မှာ RE အကြောင်းရင်းပြတုန်းက ပြောပြခဲ့သလိုပဲ \$ အမှတ်အသားကတော့ စာကြောင်းအဆုံးကို ကိုယ်စားပြုပါတယ်။ input လုပ်တဲ့ စာကြောင်းတိုင်းက နံပါတ်နဲ့ ဆုံးနေလို့ line1 ကနေ line6 အားလုံးကို ရိုက်ပြပေးပါလိမ့်မယ်။

In [7]:

```
echo -e line{1..6}'\n' | sed -n '/[[:digit:]]$/p'
```

```
line1
 line2
 line3
 line4
 line5
 line6
```

y/old_character_list/new_character_list/ ဆိုတဲ့ RE pattern နဲ့ စာလုံးတလုံးချင်းကို အစားထိုးတာမျိုးလည်း sed နဲ့ လုပ်လို့ရပါတယ်။ အောက်ပါ ဥပမာကို ကြည့်ပါ။ ဒီနေရာမှာ y/aeghimnorstvwy/AEGHIMNORSTVWY/ ဆိုတာက "a" ကိုတွေ့ရင် "A" နဲ့ အစားထိုးပေးပါ။ "e" ကို "E" နဲ့အစားထိုးပေးပါ။ "g" ကို "G" နဲ့အစားထိုးပေးပါ ... ဆိုပြီး parallel character list ကိုပေးပြီး sed ကို ညွှန်းထားတာဖြစ်ပါတယ်။

In [3]:

```
echo "Everything is awesome!" | sed y/aeghimnorstvwy/AEGHIMNORSTVWY/
```

```
EVERYTHING IS AWESOME!
```

အထက်ပါ ဥပမာက y/ / / ပုံစံကိုသုံးပြီး "Everything is awesome!" စာကြောင်းထဲမှာပါတဲ့ အင်္ဂလိပ်စာလုံးအသေးတွေကို စာလုံးအကြီးဖြစ်ပြောင်းပြထားတာဖြစ်ပါတယ်။ တကယ်က y/ / / ပုံစံကို မသုံးပဲ substitute လုပ်တဲ့ RE pattern ဖြစ်တဲ့ s/ / / နဲ့လုပ်တာက ပိုပြီးတော့ကောင်းပါတယ်။ အောက်ပါ sed command ဥပမာကို ကြည့်ပါ။ ဒီနေရာမှာ "\U" က (uppercase letter) ကိုပြောတာပါ။ နောက်ပြီး

တော့ "&" ကတော့ matched ဖြစ်တဲ့စာလုံးကို ညွှန်းတာပါ။ နောက်ဆုံးစာလုံး "g" ကတော့ ပထမဆုံးတွေ့တဲ့ စာလုံးတစ်လုံးထဲကို ပြောင်းပေးပြီး ရပ်မသွားပါနဲ့ စာကြောင်းတကြောင်းလုံးအစကနေ အဆုံးတောက်လျှောက် match ဖြစ်တိုင်းမှာ အစားထိုးပေးပါ။ globally လုပ်ပေးပါလို့ ဆိုလိုပါတယ်။

In [9]:

```
echo "Everything is awesome!" | sed 's/./\U&/g'
```

```
EVERYTHING IS AWESOME!
```

ဒီတစ်ခါတော့ uppercase letter ကို lowercase letter အဖြစ်ပြောင်းကြည့်ရအောင်။

In [11]:

```
echo "EVERYTHING IS AWESOME!" | sed 's/./\L&/g'
```

```
everything is awesome!
```

နောက် ဥပမာတစ်ခုအနေနဲ့ character class နှစ်ခုဖြစ်တဲ့ "[[:lower:]]" နဲ့ "[[:upper:]]" ကိုသုံးပြီး စာလုံးအသေးကိုတွေ့ရင် အကြီးပြောင်း၊ စာလုံးအကြီးကိုတွေ့ရင် စာလုံးအသေးကို ပြောင်းတာကို လုပ်ကြည့်ရအောင်။

In [79]:

```
echo "Everything is awesome!" | sed -E 's/([[:lower:]])|([[:upper:]])/\U\1\L\2/g'
```

```
eVERYTHING IS AWESOME!
```

Programming လောကမှာ variable name, function name တွေကို စာလုံးအကြီး၊ စာလုံးအသေး လုပ်တာကို သုံးတဲ့အသုံးအနှုန်းသတ်မှတ်ချက်ရှိပါတယ်။ မြင်သာအောင် ရေးပြရရင်၊ UPPERCASE, lowercase, Propercase နဲ့ CamelCase ဆိုပြီးသုံးပါတယ်။ စာလုံးအားလုံးကို စာလုံးအကြီးအဖြစ်ပြောင်းတာကို UPPERCASE၊ စာလုံးအားလုံးကို အသေးလုပ်တာကို lowercase၊ စာလုံးတစ်လုံးမှာ ပထမဆုံးအက္ခရာကိုပဲ စာလုံးအကြီးပြောင်းတာကို Propercase၊ စာလုံးတစ်လုံးထက်မကတာကို အတူတူတွဲကပ်ထားတဲ့ အခြေအနေမျိုးမှာ စာလုံးတစ်လုံးစီရဲ့ပထမဆုံးအက္ခရာကိုပဲ စာလုံးအကြီးပြောင်းတာကို CamelCase လုပ်တယ်လို့ ပြောကြပါတယ်။

ဟုတ်ပြီ။ sed command ကိုသုံးပြီး ဝင်လာတဲ့ စာကြောင်းကို Propercase လုပ်ခိုင်းကြည့်ရအောင်။

In [128]:

```
echo "Everything is awesome!" | sed 's/\<\(\w\)\(\S*\)/\u\1\L\2/g'
```

```
Everything Is Awesome!
```

အထက်ပါ Propsercase လုပ်တဲ့ RE မှာ "\w" က စာလုံးတစ်လုံး လို့ဆိုလိုပါတယ်။ ပြီးတော့ "\S*" ကတော့ zero or more non-whitespace characters ကိုဆိုလိုပါတယ်။ ရှေ့က Regular Expression အကြောင်းပြောတုန်းက ကျွန်တော်ရင်းပြခဲ့သလို၊ non-whitespace character ဆိုတာက ' ' (space), \r (carriage return), \n (newline), \t (horizontal tab), \v (vertical tab), \f (form feed) တို့ မဟုတ်တဲ့ စာလုံးကို ပြောတာပါ။ အဲဒါကြောင့် အင်္ဂလိပ်အက္ခရာအားလုံးအကျုံးဝင်ပါတယ်။ whitespace စာလုံးတစ်လုံးကိုတွေ့တာနဲ့ ဘောင်ထဲကိုမဝင်တော့ပါဘူး။ ဥပမာ "Everything " ကိုပဲ RE ဖြစ်တဲ့ "\<(\w)(\S)*" *နဲ့တိုက်စစ်မယ်ဆိုရင်၊ (\w) နဲ့ကိုက်တာက "E" ဖြစ်ပြီးတော့၊ (\S) နဲ့ကိုက်ညီတာကတော့* "verything" (space မပါတာကို သတိပြုပါ) ဖြစ်သွားပါလိမ့်မယ်။

အစားထိုးပေးမဲ့ အပိုင်းဖြစ်တဲ့ "/\u\1\L\2/" ကိုရှင်းပြပါမယ်။

ဒီနေရာမှာ \u\1 က စာလုံးတစ်လုံးရဲ့ပထမဆုံး အက္ခရာကို uppercase လုပ်ပေးပါလိမ့်မယ်။
\1 ဆိုတာက backreference RE ဖြစ်ပြီး ပထမဆုံး round bracket "()" ထဲက RE နဲ့ကိုက်ညီပြီးဝင်လာတဲ့ စာလုံးကို ညွှန်းတာပါ။ "(\w)" ကိုညွှန်းတာပါ။

\L\2 ဆိုတာက 2nd backreference ဖြစ်တဲ့ "\2" ကို စာလုံးအသေးလုပ်ပေးပါလို့ ခိုင်းထားတာပါ။
ဒီနေရာမှာ \2 က (\S*) နဲ့ သွားညီပါလိမ့်မယ်။

နောက်ပြီးတော့ "\<" က စာလုံးတစ်လုံးရဲ့အစboundary ကိုဆိုလိုပါတယ်။

အမှန်တကယ်က "\<(\w)(\S*)" ကို မသုံးပဲ "/\<(\w)(\w*)>/" ကိုသုံးရင်လည်း ရပါတယ်။ သို့သော် "/\<(\w)(\w*)>/" ကိုသုံးရင် "I'll" (သို့) "you'll" (သို့) "she's" စတဲ့စာလုံးတွေကို RE က စာလုံးတစ်လုံးအနေနဲ့ မမြင်တဲ့ (non-word ဆိုပြီးမြင်တဲ့) ပြဿနာရှိပါတယ်။

Bracketing လုပ်ပြီး backreference သုံးတာကို အပေါ် က ဥပမာမှာလည်း ပါခဲ့ပေမဲ့၊ ပိုပြီးနားလည်ရလွယ်အောင် နောက်ထပ် ဥပမာတစ်ခုနဲ့ ပြပါမယ်။ အောက်ပါစာကြောင်းက ပိုပြီး ကြည့်ရတာ မြင်သာပါလိမ့်မယ်။

In [39]:

```
echo "I am a NLP researcher." | sed 's/.*\(NLP\).*/\1/'
```

```
NLP
```

& သုံးတာကိုလည်း နောက်ထပ် ဥပမာ တစ်ခုနဲ့ သုံးပြပါမယ်။
အင်္ဂလိပ်စာလုံး အကြီး IS ကိုတွေ့ရင် စာလုံးအသေးအဖြစ်ပြောင်းပေးပါလို့ သုံးပြထားတာပါ။

In [131]:

```
echo "Everything IS awesome! Yes, EVERYTHINGISAWESOME" | sed 's/IS/\L&/g'
```

```
Everything is awesome! Yes, EVERYTHINGisAWESOME
```

Apply changes in multiple files

sed command သုံးပုံသုံးနည်းကို ဥပမာအနေနဲ့ echo command ကနေ (သို့) ဖိုင်တစ်ဖိုင်ကနေ ဒေတာကို piping လုပ်ပြီး သုံးပြခဲ့ပေမဲ့၊ wildcard သင်္ကေတတွေ၊ တခြား linux command တွေနဲ့ sed ကို တွဲသုံးပြီး ဖိုင်တွေအများကြီး ကို ဝင်ပြင်လို့ရတယ်ဆိုတာကိုလည်း မြင်ကြမယ်လို့ ထင်ပါတယ်။

ဥပမာ လက်ရှိရောက်နေတဲ့ path မှာ fileဆိုတဲ့စာလုံးနဲ့စတဲ့ ဖိုင်တွေအားလုံးကို find command နဲ့ ရှာဖွေခိုင်းပြီး၊ အဲဒီဖိုင်တွေထဲမှာရှိတဲ့ စာကြောင်းတွေမှာ Mingalar ဆိုတာပါရင် အဲဒီစာလုံးကိုဖျက်ပေးပါဆိုတဲ့ အလုပ်မျိုးကို ခိုင်းချင်ရင် အောက်ပါအတိုင်း ခိုင်းလို့ ရပါတယ်။
wildcard သင်္ကေတ * ကိုသုံးထားတာရယ်၊ နောက်ပြီးတော့ တခြား linux command တစ်ခုဖြစ်တဲ့ xargs ဆိုတာကို တွဲသုံးထားတာကိုပါ လေ့လာစေချင်ပါတယ်။

In [87]:

```
find -type f  -name "file*" | xargs sed  's/Mingalar//g'
```

```
 bar!
I am fileA.
I was born in April.
My blood type is O.
 bar!
I am fileA.
I was born in April.
My blood type is O.
Hi!
I am fileB.
I was born in November.
My blood type is O.
```

xargs က သူ့ရှေ့က run ခဲ့တဲ့ find command ရဲ့ output ဖြစ်တဲ့ fileနဲ့စတဲ့ ဖိုင်နာမည်တွေကိုအားလုံးကို တခုချင်းစီ sed ဆီကို argument အဖြစ်ပို့ပေးတဲ့ အလုပ်ကို လုပ်ပါတယ်။ ဖိုင်နာမည်တွေကို မြင်သာအောင် find command အပိုင်းကိုပဲ run ကြည့်ရအောင်။ လက်ရှိ path နဲ့ သူရဲ့sub-folders တွေအားလုံးကိုရှာဖွေခဲ့တာမှာ fileဆိုတဲ့နာမည်နဲ့စတဲ့ ဖိုင်သုံးဖိုင် ရိတာကို အောက်ပါအတိုင်းတွေ့ရပါလိမ့်မယ်။

In [88]:

```
find -type f  -name "file*"
```

```
./folderB/fileA
./fileA
./fileB
```

sed command ရဲ့အရမ်းအသုံးဝင်တဲ့ -i option ကို လည်းမိတ်ဆက်ပါမယ်။ ဒီ option က ပေးလိုက်တဲ့ ဖိုင်ကို ဝင်ပြင်ပေးနိုင်၊ ဝင်ရေးနိုင်တဲ့ option ပါ။ အင်္ဂလိပ်လိုတော့ "edit files in place" ဆိုပြီး sed manual မှာ ရှင်းပြထားတာကို man sed နဲ့ ဝင်ကြည့်ရင် တွေ့ရပါလိမ့်မယ်။

အောက်မှာပြထားတဲ့ ဥပမာက myanmar-food.txt ဖိုင်အသစ်ကိုဆောက်ပြီး၊ အဲဒီဖိုင်ထဲက "MontHinGar" ဆိုတဲ့ စာလုံးကို "Rakhine MontDii" ဆိုတဲ့ စာလုံးနဲ့ sed command ကို -i option ပေးပြီး ဝင်ပြင်ခိုင်းပါမယ်။

In [96]:

```
echo -e "MontHinGar\nLaphet\nBainMont" > myanmar-food.txt
cat myanmar-food.txt
```

```
MontHinGar
Laphet
BainMont
```

In [97]:

```
sed -i 's/MontHinGar/Rakhine MontDii/' ./myanmar-food.txt
```

myanmar-food.txt ဖိုင်မှာ ဝင်ပြင်သွားတာကို မြင်ရအောင် cat command နဲ့ ရိုက်ပြခိုင်းကြည့်ရအောင်။

In [98]:

```
cat ./myanmar-food.txt
```

```
Rakhine MontDii
Laphet
BainMont
```

myanmar-food.txt ဖိုင်ကိုဝင်ပြင်သွားတာကို မြင်ရပါလိမ့်မယ်။

ဒီ -i option က အသုံးဝင်သလို၊ မှားသုံးမိရင် ရှိနေတဲ့ ဖိုင်ကိုဝင်ရေးမှာမို့လို့ သတိထားပြီး သုံးရပါမယ်။
ဖိုင်တစ်ဖိုင်လောက်ကို မှားပြီးဝင်ပြင်မိရင် ပြန်ပြင်လို့ရနိုင်ပေမဲ့၊ ဖိုင်တွေ ရာနဲ့ထောင်နဲ့ချီပြီး မှားပြင်မိသွားရင်တော့ လွယ်ကူတဲ့ ကိစ္စမဟုတ်ပါဘူး။
အဲဒီအတွက် ရှိနေတဲ့ ဖိုင်ကို ဝင်မပြင်ခင် backup အရင်ကူးပြီးတော့မှ ပြင်ပါဆိုပြီး sed command ကိုခိုင်းလို့ရပါတယ်။ backup ကူးပေးစေချင်ရင် -i ရဲ့နောက်မှာ backup ဖိုင်နာမည်ရဲ့suffix ကိုကပ်ပြီးရိုက်ပေးရပါတယ်။ ဥပမာ .bak ဆိုတဲ့ file extension တပ်ပေးစေချင်ရင် -i".bak" ဆိုပြီးတော့ ရိုက်ပေးရပါမယ်။ အောက်ပါ command ကို ကြည့်ပါ။

In [123]:

```
sed -i".bak" 's/^/>\t/' ./myanmar-food.txt
```

sed command က backup ကူးပေးတဲ့ ဖိုင်ကိုဖိုင်ရော၊ နဂိုဖိုင်ကိုကော ls နဲ့ ရာကြည့်ရအောင်။
အောက်ပါအတိုင်း myanmar-food.txt.bak ဖိုင်အသစ်ကိုတွေ့ရပါလိမ့်မယ်။

In [125]:

```
ls myanmar-food.txt*
```

```
myanmar-food.txt  myanmar-food.txt.bak
```

's/^/>\t/' က စာကြောင်းတိုင်းရဲ့အစ (^)မှာ ">" သင်္ကေတနဲ့ TAB တစ်ခုကို အစားထိုးခိုင်းထားလို့၊ အဲဒီအတိုင်း myanmar-food.txt ဖိုင်မှာ ဝင်ပြင်ပေးသွားတာကို တွေ့ရပါလိမ့်မယ်။

In [126]:

```
cat myanmar-food.txt
```

```
>       Rakhine MontDii
>       Laphet
>       BainMont
```

နဂို အော်ရဂျင်နယ် ဖိုင်ကိုတော့ myanmar-food.txt.bak ဆိုတဲ့နာမည်နဲ့ သိမ်းထားတာကို အောက်ပါအတိုင်း တွေ့ရပါလိမ့်မယ်။

In [127]:

```
cat myanmar-food.txt.bak
```

```
Rakhine MontDii
Laphet
BainMont
```

sed command ကို သုံးတတ်ရင် သုံးတတ်သလိုပါပဲ။ Text processing မှာ အများကြီး အထောက်အကူဖြစ်ပါလိမ့်မယ်။ စသုံးစမှာ တော့ ခက်ချင်ခက်ပါလိမ့်မယ်။ ဒါပေမဲ့ sed command ရဲ့ syntax ကိုသိထားပြီး၊ အသုံးများတဲ့ command တွေဖြစ်တဲ့ d (delete), i (insert), a (append), c (change) စတဲ့ command တွေကို နားလည်ရင်ကို တော်တော်လေးကို အလုပ်လုပ်ရတာ အဆင်ပြေသွားပါလိမ့်မယ်။ sed command systax တွေကိုတော့ အကြမ်းမျဉ်းအားဖြင့် အောက်ပါပုံစံအတိုင်း ကျွန်တော်ကတော့ မှတ်သားထားပါတယ်။

1. [address] command
2. [line-address] command
3. address{
   command1
   command2
   command3
   }
4. address{command1; command2; command3}
5. [address]s/pattern/replacement/flags
6. [address]y/abc/xyz/
7. sed [options] commands [file-to-edit]

နံပါတ်၃ ပုံစံ လိုရေးမယ်ဆိုရင် သတိထားရမဲ့ အချက်တွေရှိပါတယ်။
နောက်ဆုံး တွန့်ကွင်း (}) ကို သပ်သပ်စာကြောင်းမှာ ရေးရပါမယ်။ command တွေရဲ့ ရှေ့မှာ space ကီး၊ TAB ကီး စတာတွေနဲ့ indentation လုပ်တာကို ခွင့်ပြုပေမဲ့၊ command ရဲ့နောက်မှာ space ပိုရိုက်မိတာမျိုးကို sed syntax မှာ မရပါဘူး။ Error message ပေးပါလိမ့်မယ်။ sed က ကျွန်တော်တို့လုပ်ခိုင်းထားတဲ့ command ကို နားမလည်တော့ဘူးဆိုရင် "Command garbled" ဆိုတဲ့ error message ကို ပေးပါလိမ့်မယ်။ နံပါတ်၃ ပုံစံမျိုးမဟုတ်ပဲ command အားလုံးကို တစ်ကြောင်းထဲမှာ စုရေးတဲ့ ပုံစံဖြစ်တဲ့ နံပါတ်၄ ပုံစံနဲ့သုံးရင်လည်း အတူတူပါပဲ။ ဖတ်ရတာ လွယ်ကူအောင်လို့ တခါတလေမှာ နံပါတ်၃ ပုံစံကို အသုံးပြုကြတာပါ။ sed command တွေကိုစုရေးထား

ပြီး ဖိုင်ထဲမှာ သိမ်းထားတဲ့ အခါမှာ နောက်ပြန်ကြည့်ရင် ဘာတွေလုပ်ထားခဲ့သလဲဆိုတာကို ပြန်သတိရဖို့အတွက် programming language တွေမှာ လုပ်ကြသလို comment ရေးထားလို့ရပါတယ်။ comment ရေးမယ်ဆိုရင်တော့ စာကြောင်းရဲ့ရှေ့မှာ # ကိုခံပြီး ရေးပါတယ်။ ဥပမာ # This is a comemnt of sed.

နံပါတ်၃ syntaxပုံစံမျိုး sed command တွေကို ဖိုင်ထဲမှာ သိမ်းထားမယ်။ အဲဒီဖိုင်ကို sed command ဆီကို -f option နဲ့ pass လုပ်ပြီး၊ အလုပ်လုပ်ပေးစေချင်တဲ့ ဖိုင်တစ်ဖိုင်အပေါ်မှာ text processing လုပ်တာကို ရှင်းပြပါမယ်။

အရင်ဆုံး sed command တွေကို သိမ်းထားတဲ့ sed.check ဖိုင်ကို cat လုပ်ကြည့်ရအောင်။

In [135]:

```
cat sed.check
```

```
/^[0-9]/{
p
s/^[[:digit:]]\+/(&)/
p
}
```

sed.check ဖိုင်ထဲက စာကြောင်းတစ်ကြောင်းချင်းစီကို ရှင်းပြပါမယ်။

/^[0-9]/{	ဂဏန်းနဲ့စတဲ့ စာကြောင်းတွေကို ညွှန်းပါတယ်။ အဲဒါကြောင့် ပထမဆုံးစာကြောင်းကို ကျော်သွားပါလိမ့်မယ်။
p	ဂဏန်းနဲ့စတဲ့ စာကြောင်းက pattern space ဆီဝင်လာရင် အဲဒီစာကြောင်းကို မော်နီတာမှာ ရိုက်ထုတ်ပြခိုင်းတာပါ။
s/^[[:digit:]]+/(&)/	စာကြောင်းထိပ်ဆုံး မှာရှိတဲ့ ဂဏန်းကို ကွင်းစ၊ ကွင်းပိတ်ထဲကို ထည့်ရေးခိုင်းတာပါ။
p	pattern space မှာရှိနေတဲ့ စာကြောင်းကို မော်နီတာမှာ ရိုက်ထုတ်ခိုင်းတာပါ။
}	command တွေမစခင်မှာ open brace "{" ဖွင့်ခဲ့သလို၊ command တွေအဆုံးမှာ close brace "}"နဲ့ပိတ်တာပါ။

item-list ဖိုင်ကို sed.check ဖိုင်နဲ့ ပြင်ကြည့်ရအောင်။
အောက်ပါအတိုင်း တွေ့ရပါလိမ့်မယ်။

In [136]:

```
sed -n -f ./sed.check ./item-list
```

```
1       pencil  10
(1)     pencil  10
2       ruler   1000
(2)     ruler   1000
3       eraser  42
(3)     eraser  42
```

output ကိုကြည့်ပြီး နားလည်မယ်လို့ ထင်ပါတယ်။
မပြင်ခင်မှာ တခေါက်၊ ပြင်ပြီးတော့ နောက်တစ်ခေါက် မော်နီတာမှာ စုစုပေါင်း နှစ်ခါ ရိုက်ထုတ်ခိုင်းခဲ့တာပါ။
ပြင်တာကလည်း စာကြောင်းရဲ့ထိပ်ဆုံးမှာရှိနေတဲ့ နံပါတ်တွေကို ကွင်းစ ကွင်းပိတ်နဲ့ ရေးတဲ့ပုံစံအဖြစ်ပြောင်းရေးခိုင်းတာပါ။

sed ရဲ့option တွေကို အသေးစိတ်သိချင်ရင် info sed ကို ဖတ်ပါ။ man sed ထက် ပိုပြည့်စုံပါတယ်။
ကျွန်တော် ညွှန်းထားတဲ့ Reference link တွေကိုလည်းဖတ်ကြည့်ပါ။

31. awk

In [6]:

```
echo "UPPER" | awk '{print tolower($0)}'
```

```
upper
```

In [55]:

```
echo -n "Who are you?" | xxd -b
```

```
00000000: 01010111 01101000 01101111 00100000 01100001 01110010  Who a
r
00000006: 01100101 00100000 01111001 01101111 01110101 00111111  e yo
u?
```

In [65]:

```
echo -n "Who are you?" | xxd -b | awk '{print $2, $3, $4;exit;}'
```

```
01010111 01101000 01101111
```

print common line from two file

In [4]:

```
awk 'NR==FNR{a[$0];next} $0 in a' fileA fileB
```

```
My blood type is O.
```

Suppose we have a data file like this

20081010 1123 xxx 20081011 1234 def 20081012 0933 xyz 20081013 0512 abc 20081013 0717 def ...thousand of lines... where "xxx", "def", etc. are operation codes. We want to replace each operation code with its description. We have another file that maps operation codes to human readable descriptions, like this:

abc withdrawal def payment xyz deposit xxx balance ...other codes... We can easily replace the opcodes in the data file with this simple awk program, that again uses the two-files idiom:

use information from a map file to modify a data file awk 'NR==FNR{a[$1] =2;next} {$3 = $a[3]}1' mapfile datafile

Ref: http://www.catonmat.net/blog/ten-awk-tips-tricks-and-pitfalls/ (http://www.catonmat.net/blog/ten-awk-tips-tricks-and-pitfalls/)
I want to think another example based on following (e.g., grouping based on publication year of paper, or adding summarization ... total no. of papers based on year at the end of file):
awk 'NR==FNR{if($0 > max)max =0;next} {$0 = max−0}1' numbers.txt numbers.txt

In [9]:

```
awk '/Mingalar/,/April/' ./fileA
```

```
Mingalar bar!
I am fileA.
I was born in April.
```

In [10]:

```
awk '/Mingalar/,/April/{if (!/Mingalar/&&!/April/)print}' ./fileA
```

```
I am fileA.
```

In [11]:

```
awk '/Mingalar/,/April/{if (!/Mingalar/)print}' ./fileA
```

```
I am fileA.
I was born in April.
```

In [12]:

```
awk '/Mingalar/,/April/{if (!/April/)print}' ./fileA
```

```
Mingalar bar!
I am fileA.
```

In [13]:

```
awk '/April/{p=0};p;/Mingalar/{p=1}' ./fileA
```

```
I am fileA.
```

In [14]:

```
awk '/April/{p=0} /Mingalar/{p=1} p' ./fileA
```

```
Mingalar bar!
I am fileA.
```

In [15]:

```
awk 'p; /April/{p=0} /Mingalar/{p=1}' ./fileA
```

```
I am fileA.
I was born in April.
```

In [16]:

```
awk '/Mingalar/{p=1};p;/April/{p=0}' ./fileA
```

```
Mingalar bar!
I am fileA.
I was born in April.
```

In [17]:

```
awk '{ print NR, $0}' ./fileA
```

```
1 Mingalar bar!
2 I am fileA.
3 I was born in April.
4 My blood type is O.
```

In [18]:

```
awk 'END {print NR}' ./fileA
```

```
4
```

In [20]:

```
awk 'NR==2' ./fileA
```

```
I am fileA.
```

In [25]:

```
awk '{print $NF}' ./fileA
```

```
bar!
fileA.
April.
O.
```

In [26]:

```
awk '{field = $NF} END {print field}' ./fileA
```

```
O.
```

In [28]:

```
awk 'NF > 3' ./fileA
```

```
I was born in April.
My blood type is O.
```

In [29]:

```
awk '$NF > 4' ./fileA
```

```
Mingalar bar!
I am fileA.
I was born in April.
My blood type is O.
```

In [31]:

```
awk '{nf = nf + NF} END{print nf}' ./fileA
```

```
15
```

I should find/read relating to p flag of awk for explaning in Myanmar language

sed raw data
awk structure data

32. tac (concatenate and print files in reverse)

ဖိုင်ထဲမှာရှိတဲ့ စာကြောင်းတွေကို ဖိုင်ရဲ့နောက်ဆုံးစာကြောင်းကနေ ထိပ်ဆုံးအကြောင်းထိ ပြောင်းပြန်ပြန်ရိုက်ပေးတဲ့ command ပါ။ (တနည်းအားဖြင့် ရှေ့ပိုင်းမှာ လေ့လာခဲ့တဲ့ cat command ရဲ့ ဆန့်ကျင်ဘက် အလုပ်ကို လုပ်ပေးပါတယ်)

In [9]:

```
tac fileA
```

```
My blood type is O.
I was born in April.
I am fileA.
Mingalar bar!
```

အောက်ပါအတိုင်း fileA ကို cat နဲ့ ရိုက်ကြည့်ပြီး tac command ရဲ့ output နဲ့ နှိုင်းယှဉ်ကြည့်ရင် ဘယ်လိုကွာသလဲဆိုတာကို မြင်ပါလိမ့်မယ်။

In [10]:

```
cat fileA
```

```
Mingalar bar!
I am fileA.
I was born in April.
My blood type is O.
```

33. rev (reverse lines characterwise)

rev command က ဖိုင်ထဲမှာ ရှိတဲ့ စာကြောင်း တစ်ကြောင်းချင်းစီကို ပြောင်းပြန်ပြန်ရိုက်ပေးတဲ့ အလုပ်ကို လုပ်ပေးပါတယ်။

ရှေ့မှာ အကြိမ်ကြိမ် သုံးခဲ့တဲ့ fileA ကို rev command ကို သုံးပြီး ပြောင်းပြန်ပြန်ရိုက်ခိုင်းကြည့်ရအောင်။

In [11]:

```
rev fileA
```

```
!rab ralagniM
.Aelif ma I
.lirpA ni nrob saw I
.O si epyt doolb yM
```

34. paste (merge lines of files)

ဖိုင်တွေထဲမှာရှိတဲ့ စာကြောင်းတွေကို ပေါင်းပြီး ရိုက်ထုတ်ချင်တဲ့ အခါ၊ ပေါင်းပြီး သိမ်းချင်တဲ့ အခါတွေမှာ အသုံးပြုပါတယ်။ လက်တွေ့မှာ အရမ်းကို အသုံးဝင်တဲ့ command တစ်ခု ဖြစ်ပါတယ်။

ဥပမာ fileA နှင့် fileB ကို ပေါင်းပြီး ရိုက်ထုတ်ပေးခိုင်းချင်တဲ့ အခါမှာ အောက်ပါအတိုင်း paste command ကို အသုံးပြုနိုင်ပါတယ်။

In [12]:

```
paste fileA fileB
```

```
Mingalar bar!   Hi!
I am fileA.     I am fileB.
I was born in April.    I was born in November.
My blood type is O.     My blood type is O.
```

ဘာ parameter မှ မပေးဘူးဆိုရင် paste command က ဖိုင်တစ်ဖိုင်နဲ့ တစ်ဖိုင်ကြားကို tab နဲ့ ခြားပေးပါလိမ့်မယ်။ တကယ်လို့ ကိုယ်က ခြားပေးစေချင်တဲ့ စာလုံးကို -d (သို့) --delimiters ဆိုတဲ့ parameter နဲ့ ညွှန်ကြားပေးရင်၊ အဲဒီညွှန်ကြားပေးတဲ့ parameter နဲ့ခြားပြီး မော်နီတာ စကရင်မှာ ရိုက်ထုတ်ပေးပါလိမ့်မယ်။

ဥပမာ fileA နှင့် fileB အကြားက စာကြောင်းတွေကို pipe စာလုံး ("|") နဲ့ ခြားပေးစေချင်ရင် အောက်ပါအတိုင်း command ပေးလို့ ရပါတယ်။

In [13]:

```
paste -d'\|' fileA fileB
```

```
Mingalar bar!|Hi!
I am fileA.|I am fileB.
I was born in April.|I was born in November.
My blood type is O.|My blood type is O.
|
```

အထက်ပါ output မှာ နောက်ဆုံးစာကြောင်းက pipe တစ်ခုပဲ မြင်နေရတာက၊ fileB မှာ စာဘာမှ မရိုက်ထားပဲ enter ခေါက်ထားခဲ့တဲ့ စာကြောင်းရှိလို့ပါ။

နောက်ထပ် ဥပမာ တစ်ခုအနေနဲ့ fileA နှင့် fileB အကြားက စာကြောင်းတွေကို စာကြောင်းတစ်ကြောင်းစီအနေနဲ့ အလှည့်ကျ ရိုက်ထုတ်ပေးချင်တဲ့ အခါမှာ အောက်ပါအတိုင်း အသုံးပြုပါတယ်။

In [14]:

```
paste -d'\n' fileA fileB
```

```
Mingalar bar!
Hi!
I am fileA.
I am fileB.
I was born in April.
I was born in November.
My blood type is O.
My blood type is O.
```

In [41]:

```
for i in {1..5}; do echo "This is line $i."; done
```

```
This is line 1.
This is line 2.
This is line 3.
This is line 4.
This is line 5.
```

In [34]:

```
find ~/Downloads/ -type f -name *.tar.gz -size +10M -exec ls -l {}\;
```

```
find: missing argument to `-exec'
```

In [99]:

```
#!/bin/bash

for file in ./test/accessory-box/*  ./test/box/*  ./test/cup/*  ./test/emo/*  ./tes
do

filestr=`basename $file`

train_file=`echo $file | sed 's/test/train/'`
dirstr=`dirname $train_file`

if [ `ls $dirstr | grep -c $filestr` -eq 1 ]; then

   echo "$file exists in $dirstr"
   #ls $dirstr | grep $filestr
   ls $train_file
   #rm $train_file
fi
done
```

```
ls: cannot access './train/accessory-box': No such file or directory
grep: data1: Is a directory
grep: data2: Is a directory
grep: data3: Is a directory
grep: data4: Is a directory
grep: data5: Is a directory
grep: folderA: Is a directory
grep: folder-athit: Is a directory
grep: folderB: Is a directory
grep: folder-new: Is a directory
grep: screen: Is a directory
grep: wildcard: Is a directory
bash: [: too many arguments
ls: cannot access './train/box': No such file or directory
grep: data1: Is a directory
grep: data2: Is a directory
grep: data3: Is a directory
grep: data4: Is a directory
grep: data5: Is a directory
grep: folderA: Is a directory
```

soffice --convert-to jpg ./fileA

for req in (*catrequirements.txt*); *dopipinstall*req; done

#!/bin/bash

for file in *.txt; do

exit if there are no .txt files

if [! -f $file]; then exit fi

b=basename $file .txt

soffice --convert-to jpg $b.txt 2> /dev/null echo ""

done

Note: example of using apt, apt-get and showing how linux is good!

sudo apt install graphviz

(py2.7.13) lar@lar-air:~/linux-cmd$ cat process.gv digraph G { subgraph cluster0 { node [style=filled,color=white]; style=filled; color=lightgrey; a0 -> a1 -> a2 -> a3; label = "process #1"; } subgraph cluster1 { node [style=filled]; b0 -> b1 -> b2 -> b3; label = "process #2"; color=blue } start -> a0; start -> b0; a1 -> b3; b2 -> a3; a3 -> a0; a3 -> end; b3 -> end; start [shape=Mdiamond]; end [shape=Msquare]; }

dot -Tpdf ./process.gv > process.pdf dot -Tpng ./process.gv > process.png

Note: ls -ltr, gzip, bzip2, unzip, zless, tar, shutdown, ps, free, top, df, kill, chmod, chown, passwd, whereis, locate, man, sudo, apt-get, date, cal, wget, echo, env, export, ifconfig, ping, tee, fg, bg, jobs, whoami, nautilus, ln unalias, uniq, sort, source, vi, emacs, history, tr, at, w, set (to show your current enviroment in zsh or bash), display, eog, fmt convert

dos2unix (I am not telling)

How to sort a file by a column

Columns are separated by a space, we sort numerically (-n) and we sort by the 10'th column (-k10) bash does the job here, no perl needed ;)

sort -t' ' -n -k10 eSet1_both.txt

======

sudo apt-get install tree tree -d -L 1

. ├── bin ├── boot ├── cdrom ├── data1 ├── data2 ├── dev ├── etc ├── home ├── lib ├── lib64 ├── lost+found ├── media ├── mnt ├── opt ├── proc ├── root ├── run ├── sbin ├── srv ├── sys ├── tmp ├── usr └── var

uname -a lsb_release -a cat /etc/lsb-release cat /etc/debian_version

Wild Card Regular Expression Directives such as > | >> &

Some useful line editing key bindings provided by the Readline library:

Ctrl-A: go to the beginning of line
Ctrl-E: go to the end of line
Alt-B: skip one word backward
Alt-F: skip one word forward
Ctrl-U: delete to the beginning of line
Ctrl-K: delete to the end of line
Alt-D: delete to the end of word

Ctrl + a – go to the start of the command line
Ctrl + e – go to the end of the command line
Ctrl + k – delete from cursor to the end of the command line
Ctrl + u – delete from cursor to the start of the command line
Ctrl + w – delete from cursor to start of word (i.e. delete backwards one word)