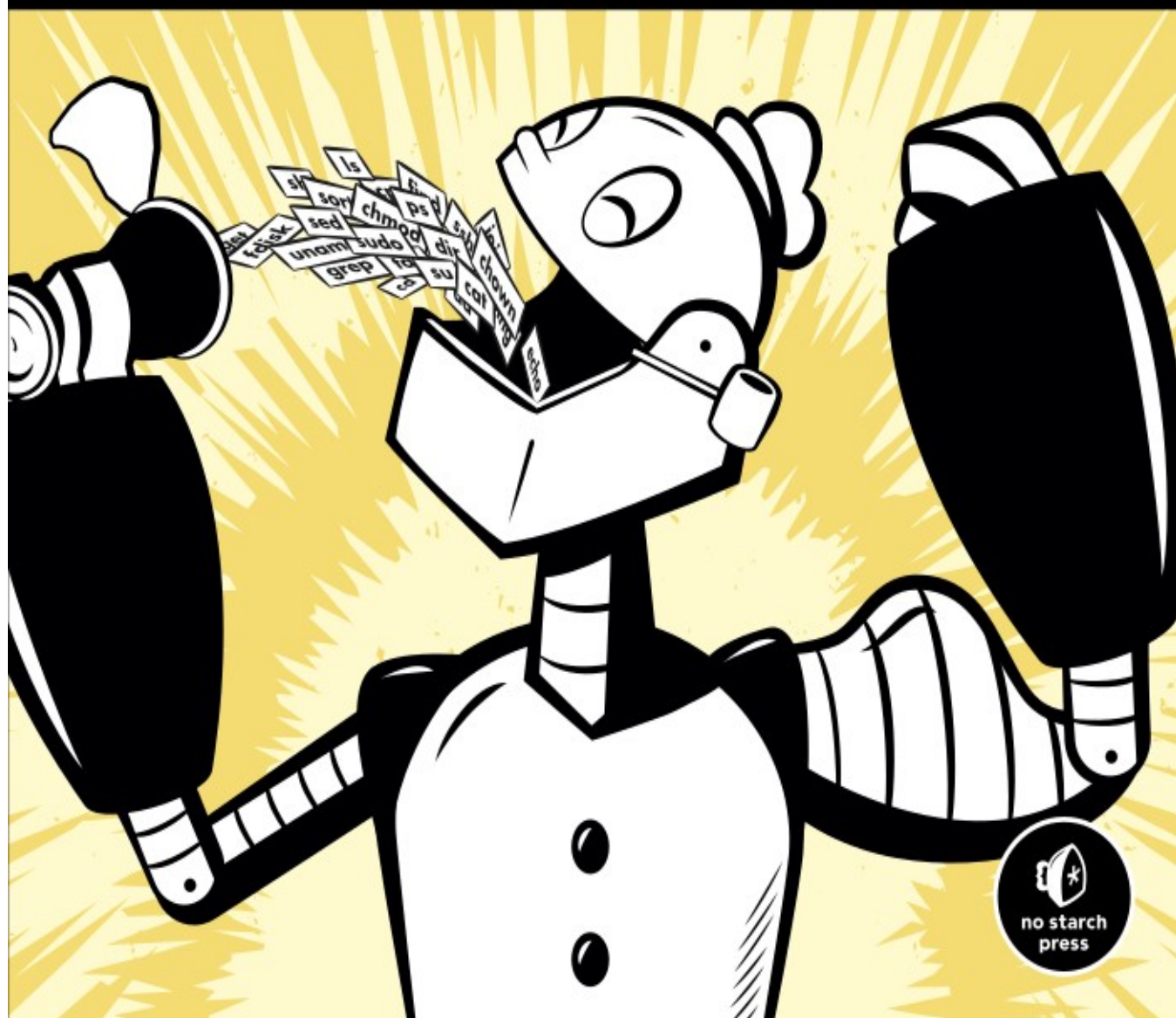


THE LINUX COMMAND LINE

A COMPLETE INTRODUCTION

WILLIAM E. SHOTTS, JR.



ဘာသာပြန်သူ၏အမှာစာ

ယခုစာအုပ်မှာ William E. Shotts, Jr. ရေးသားခဲ့သည့် The complete linux command line *A complete introduction* စာအုပ်ကို ယခုမှစတင်ပြီး linux ကို စတင်လေ့လာမည့်သူများအတွက် တဖက်တလမ်းမှ အကူအညီရစေချင်သောကြောင့် ကျွန်တော်ဉာဏ်မှီသမျှ ဘာသာပြန်ထားခြင်းဖြစ်ပါတယ်။ ။

ကျွန်တော်က Professional ICT Technician တယောက်မဟုတ်သလို Professional Translator တယောက်လည်းမဟုတ်ပါဘူး။ ။ ဝါသနာအရ လုပ်တာပါ။ ။ ဒါကြောင့်အမှားအယွင်းများရှိခဲ့လျှင်လည်းကျွန်တော့်ရဲ့ ညံ့ဖျင်းမှုသာဖြစ်ပြီးတော့မူရင်းစာရေးဆရာနဲ့မသက်ဆိုင်ပါကြောင်းတောင်းပန်အပ်ပါတယ်။ ။ ကွင်းစကွင်းပိတ် (...) နှစ်ခုကြားမှာရေးထားတဲ့စာတွေသည်ကျွန်တော့်အဘော်သာဖြစ်ပြီးမူရင်းစာရေးသားသူရဲ့အဘော်မဟုတ်ပါ။ ။ မူရင်းစာအုပ်မှာမူရင်းစာရေးသားသူက Ubuntu version အဟောင်းကိုအသုံးပြုပြီးပြထားတာဖြစ်သော်လည်းကျွန်တော့်စက်မှာ Ubuntu 20.04 (64 bits) ထည့်သွင်းထားတာကြောင့် အဲဒါနဲ့ပဲ example တွေကိုလုပ်ပြသွားမှာဖြစ်တဲ့အတွက်အချို့ command တွေမှာမူရင်းနဲ့ကွဲလွဲနေတာမျိုး ဒါမှမဟုတ် လက်ရှိခေတ် distro တွေပေါ်မှာ ပယ်ဖျက်လိုက်တဲ့ ပေးမသုံးတော့တဲ့ command တွေရှိခဲ့မယ်ဆိုရင် ချန်လှပ်ထားခဲ့မှာမို့ နားလည်ပေးဖို့ပြောလိုပါတယ်။ ။

ခင်ဗျားတို့အနေနဲ့စာရိုက်ရမှာပျင်းလို့ code တွေကို command prompt ထဲကိုယခု pdf file ထဲကနေကူးထည့်ချင်တယ်ဆိုရင် မျည်း၂ကြောင်းနဲ့ပြထားတဲ့ ကြားထဲက code တွေကိုလုပ်လို့ရပေမယ့် code ကို Bold နဲ့စာတွေရဲ့အလယ်မှာပြထားတာတွေကိုလုပ်ရင်တော့အလုပ်လုပ်မှာမဟုတ်ပါဘူး။ ။ ဘာလို့လဲဆိုတော့ မျည်း၂ကြောင်းနဲ့ပြထားတဲ့ ကြားထဲက code တွေအကုန်လုံးကို Ubuntu 20.04 ရဲ့ terminal မှာအကုန်လုံးစမ်းပြီးအလုပ်လုပ်တယ်ဆိုမှ ကျွန်တော်ကထည့်ပေးထားတာဖြစ်ပြီးတော့ Bold နဲ့စာတွေ ကတော့ font အရကြည့်ကောင်းအောင် space တွေထည့်ပြီးလုပ်ထားရတာဖြစ်လို့ပါပဲ။ ။

နောက်တခုက မူရင်းစာအုပ်မှာ youtube link တွေမပါဝင်ပါဘူး။ ။ ကျွန်တော်ကပိုပြီးပြည့်ပြည့်စုံစုံသိစေလိုတဲ့စေတနာနဲ့ထည့်ပေးထားတာဖြစ်ပါတယ်။ ။

မူရင်းစာရေးသူရဲ့ခွင့်ပြုချက်ကိုအခက်အခဲအမျိုးမျိုးကြောင့်မတောင်းခံနိုင်ခဲ့တဲ့အတွက်ယခုအမှာစာကနေပဲလေးစားစွာ Credit ပေးပါတယ်။ ။

နောက်ပီးယခုစာအုပ်ကိုဝါသနာတူသူများအချင်းချင်းအခမဲ့ကူးယူဖြန့်ချိနိုင်သော်လည်းစီးပွားဖြစ်ထုတ်ဝေကာရောင်းမစားကြပါရန်လေးစားစွာမေတ္တာရပ်ခံအပ်ပါတယ်ခင်ဗျာ။ ။

လေးစားစွာ

ဖြင့် ...

(၅.၄.၂၀၂၂)

INTRODUCTION

ကျွန်တော်ခင်ဗျားတို့ကို ဇာတ်လမ်းတပုဒ်ပြောပြချင်ပါတယ်။ 1991 ခုနှစ်မှာ Linus Torvalds က ပထမဦးဆုံးသော Linux kernel ကိုဘယ်လိုရေးခဲ့တယ်ဆိုတဲ့ ဇာတ်လမ်းလည်းမဟုတ်ပါဘူး။ ခင်ဗျားအနေနဲ့အဲဒီဇာတ်လမ်းကို Linux စာအုပ်ပေါင်းများစွာမှာဖတ်နိုင်ပါတယ်။ လွန်ခဲ့တဲ့ အစောပိုင်းနှစ်အနည်းငယ်က Richard Stallman (သူ့ကို RMS ဟုလည်းခေါ်ကြသည်) မှ အခမဲ့ Unix-like operating system တခုကိုဖန်တီးဖို့အတွက် GNU project ကိုဘယ်လိုစတင်ခဲ့တယ်ဆိုတဲ့ ဇာတ်လမ်းလည်းမဟုတ်ပါဘူး။ ဒါကလည်းအရေးကြီးတဲ့ဇာတ်လမ်း တခုပါပဲ ဒါပေမယ့် အခြားသော Linux စာအုပ်ပေါင်းများစွာမှာလည်းပဲ အဲဒီဇာတ်လမ်းက ပါပြီးသားဖြစ်ပါတယ်။ ကျွန်တော်ပြောချင်တဲ့ ဇာတ်လမ်းက ခင်ဗျားအနေနဲ့ခင်ဗျားရဲ့ computer ထိမ်းချုပ်မှုပိုင်းကိုဘယ်လိုပြန်လည်ရယူနိုင်မလဲဆိုတဲ့ဇာတ်လမ်းကိုပြောမှာဖြစ်ပါတယ်။

1970 ခုနှစ်အနှောင်းပိုင်းကာလများမှာ ကျွန်တော်ကောလိပ်ကျောင်းသားတယောက်အနေနဲ့ computer ကို အသုံးပြုနေတဲ့အချိန် အဲဒီမှာခေတ်ပြောင်းတော်လှန်ရေးတခုဖြစ်ပွားနေပါတယ်။ microprocessor တီထွင်နိုင်ခဲ့မှုကြောင့် ခင်ဗျားလို ကျွန်တော်လိုလူတွေအတွက်တကယ့်ကို computer တလုံးပိုင်ဆိုင်နိုင်ဖို့ဖြစ်လာခဲ့ပါတယ်။ company ကြီးတွေနဲ့အစိုးရကြီးတွေသာ computer တွေကိုသုံးနိုင်တဲ့ ကမ္ဘာကြီးရဲ့အခြေအနေကိုယခုခေတ်လူတွေ အနေနဲ့တော့စိတ်ကူးကြည့်ဖို့(နားလည်ဖို့)ခက်ခဲပါလိမ့်မယ်။ (အဲဒီခေတ်မှာ)ခင်ဗျားအနေနဲ့လုပ်နိုင်တာဘာမှသိပ်မရှိဘူးလို့ပဲဆိုကြပါစို့။

ယနေ့မှာတော့ကမ္ဘာကြီးကခြားနားသွားပါပြီ။ လက်ပတ်နာရီအသေးလေးကနေ ဧရာမ Data Center ကြီးတွေအထိ computer တွေကနေရာတကာမှာရှိနေပါပြီ။ နေရာတကာမှာတွေ့နေရတဲ့ computer တွေကိုပါထပ်ပေါင်းပြီးပြောရမယ်ဆိုရင် ကျွန်တော်တို့မှာ နေရာတကာမှာတွေ့နေရတဲ့ network တွေလည်းပဲ ရှိနေပါပြီ။ အဲဒါကတကိုယ်ရည်စွမ်းဆောင်နိုင်စွမ်းနဲ့ တီထွင်ဖန်တီးမှုဆိုင်ရာလွတ်လပ်မှုတွေရဲ့အံ့ဩသင့်ဖွယ်ရာခေတ်သစ်တခုကိုဖန်တီးပေးလိုက်ပါတယ်။ ဒါပေမယ့်လွန်ခဲ့တဲ့ဆယ်စုနှစ်များမှာတစ်ခုတခုဟာဖြစ်ပွားခဲ့ပါတယ်။ ဧရာမကော်ပိုရေးရှင်းကြီးတခုက ကမ္ဘာပေါ်မှာရှိသမျှ computer အများစုအပေါ်မှာပြစ်ဒဏ်(ဥပဒေ)သတ်မှတ်ချက်တွေနဲ့ထိန်းချုပ်ထားခဲ့ပြီးတော့ ခင်ဗျားအနေနဲ့ computer တွေနဲ့ပတ်သက်ပြီးတော့ဘာလုပ်ခွင့်ရှိတယ် ဘာလုပ်ခွင့်မရှိဘူးဆိုတာတွေကိုသူက ဆုံးဖြတ်ပေးလာခဲ့ပါတယ်။ ကံကောင်းတာကတော့ကမ္ဘာအရပ်ရပ်မှာရှိတဲ့လူတွေကဒီကိစ္စနဲ့ပတ်သက်ပြီးတော့တခုလုပ်လာခဲ့ပါတယ်။ သူတို့တတွေဟာ သူတို့တွေရဲ့ computer တွေအပေါ်မှာထိန်းချုပ်မှုပြန်လည်ရရှိရေးအတွက် တိုက်ပွဲဝင်ခဲ့ကြပါတယ်။ သူတို့က Linux ကိုတည်ဆောက်ခဲ့ကြပါတယ်။

လူအများစုက Linux နဲ့ပတ်သက်ပြီး Freedom အကြောင်းပြောလေ့ရှိကြပေမယ့်လူတော်တော်များများကတော့ Freedom ဆိုတာတကယ်တမ်းဘာကိုဆိုလိုတာလဲဆိုတာကိုသိလိမ့်မယ်လို့ကျွန်တော်တော့မထင်ပါဘူး။ Freedom ဆိုတာ computer ကဘာကိုလုပ်ရမယ်ဆိုတာကိုဆုံးဖြတ်ပေးနိုင်တဲ့ power ဖြစ်ပြီးတော့ ဒီ power ကိုရရှိနိုင်တဲ့တခုထဲသောနည်းလမ်းကတော့ခင်ဗျားရဲ့ computer ဘာလုပ်နေတယ်ဆိုတာကိုသိမှဖြစ်မှာပါ။

Why Use the Command Line ?

ခင်ဗျားအနေနဲ့ရုပ်ရှင်တွေထဲမှာသတိထားမိလားမသိဘူး super hacker ကြီးက --- ခင်ဗျားသိတယ်မလား အဲ့ဘဲကြီးက စစ်တပ်ရဲ့အဆင့်အရမ်းအရမ်းကိုမြင့်တဲ့လုံခြုံရေးစနစ် computer ကြီးကို စက္ကန့် ၃၀ အတွင်းမှာချိုးဖောက်ခဲ့တာ --- computer ရှေ့မှာထိုင်ပြီးတော့ သူက mouse ကိုတချက်တောင်မထိပဲနဲ့ ? အဲ့ဒါကဘာကြောင့်လဲဆိုတော့ ရုပ်ရှင်ရိုက်တဲ့သူတွေ နားလည်သိမြင်လာတာက ကျွန်တော်တို့ ၊ လူသားတွေအနေနဲ့ ၊ မွေးရာပါဗီဒီစိတ်အရ computer တလုံးပေါ်မှာဘယ်အရာမဆိုဖြစ်မြောက်အောင်လုပ်နိုင်တဲ့တခုထဲသောနည်းလမ်းကတော့ keyboard ပေါ်မှာစာရိုက်ခြင်းပဲဖြစ်တယ်ဆိုတာကိုသိလို့ပါပဲ ။

ယနေ့ခေတ် computer အသုံးပြုသူတော်တော်များများဟာ Graphical User Interface (GUI)(ဂူအီ လို့အသံထွက်တယ်) တခုထဲနဲ့သာရင်းနှီးပြီးတော့ Command Line Interface (CLI) ဟာ အတိတ်ခေတ်ကကြောက်စရာကောင်းတဲ့ အရာကြီးတခုအဖြစ် computer ရောင်းတဲ့သူတွေနဲ့ ဆရာဆရာကြီးတွေရဲ့ သင်ကြားမှုကိုခံခဲ့ရပါတယ် ။ အဲ့ဒါက ကံဆိုးတာပါပဲ ။ ဘာဖြစ်လို့လဲဆိုတော့ ကောင်းမွန်တဲ့ Command Line Interface တခုဟာ computer တလုံးကိုဆက်သွယ်ရာမှာလူသားတွေစာရေးပြီးဆက်သွယ်တဲ့နည်းလမ်းလိုမျိုးအံ့ဩဖွယ်ကောင်းလောက်အောင်ထိထိမိမိ နည်းလမ်းနဲ့ ဆက်သွယ်ပေးနိုင်တာကြောင့်ဖြစ်ပါတယ် ။ ပြောကြတာကတော့ graphical user interface က လွယ်ကူတဲ့အလုပ်တွေကိုလွယ်ကူအောင်လုပ်ပေးနိုင်ချိန်မှာ Command Line Interface က ခက်ခဲတဲ့အလုပ်တွေကိုဖြစ်မြောက်အောင်လုပ်ပေးတယ် လို့ဖြစ်ပြီးတော့ အဲ့ဒါက ယနေ့အချိန်အထိအရမ်းကိုမှန်နေတုန်းပါပဲ ။

Linux ဟာ Unix family ရဲ့ operating system နောက်ကနေ (အတုယူပြီး) လိုက်ပြီး model ထုတ်တယ်လို့ဆိုကတည်းက Unix လိုပဲတူညီတဲ့ command line tool တွေရဲ့ကြွယ်ဝတဲ့အမွေကိုမျှဝေဆက်ခံထားတာပဲဖြစ်ပါတယ် ။ 1980 ခုနှစ်အစောပိုင်းကာလတွေမှာ graphical user interface တွေကျယ်ကျယ်ပြန့်ပြန့်မွေးစားအသုံးမပြုခင်အချိန်က Unix ဟာထင်ပေါ်လာပြီးတော့ (သူကဆယ်စုနှစ်တခုစာကြိုတင်ပြီးတည်ဆောက်ပြီးသားဖြစ်သော်လည်း) အကျိုးဆက်အနေနဲ့ command line interface တွေကိုကျယ်ကျယ်ပြန့်ပြန့် ဖန်တီးလာခဲ့ကြပါတယ် ။ တကယ်တော့ အရှေ့ဦးပိုင်း Linux ကိုမွေးစားသူ(တည်ဆောက်သူ)တွေအနေနဲ့ဘာလို့ (CLI) ကိုရွေးခဲ့ကြသလဲဆိုတဲ့အားအကြီးဆုံးသောအချက်တချက်ကတော့ ၊ ဆိုကြပါစို့ ၊ windows NT ကလည်း powerful command line interface တခုဖြစ်ခဲ့ပြီးတော့ ခက်ခဲတဲ့အလုပ်တွေကိုဖြစ်မြောက်အောင်လုပ်ဆောင်နိုင်ခဲ့တာပါပဲ ။

What This Book Is About

ယခုစာအုပ်ဟာ Linux command line ပေါ်မှာ နေထိုင်နည်းကို အကျယ် တင်ပြထားခြင်းဖြစ်ပါတယ် ။ shell program, bash တို့လို program တခုထဲကိုအာရုံစိုက်ပြီးတော့ရေးထားတဲ့အခြားစာအုပ်တွေနဲ့မတူတာက ယခုစာအုပ်မှာ command line interface ပေါ်မှာဘယ်လိုအဆင်ပြေအောင်နေထိုင်ရမလဲဆိုတာကိုမြင်ကွင်းအကြီးကြီးတခုအနေနဲ့သိအောင်ပို့ဆောင်ပေးမှာဖြစ်ပါတယ် ။ ဒါတွေအားလုံးကဘယ်လိုအလုပ်လုပ်ကြတာလဲ ? အဲ့ဒါကဘာလုပ်နိုင်သလဲ ? အဲ့ဒါကိုအသုံးပြုဖို့အကောင်းဆုံးနည်းလမ်းကဘာလဲ ?

ယခုစာအုပ်ဟာ Linux system administration အကြောင်းရေးထားတာမဟုတ်ပါဘူး ။ command line နဲ့ပက်သက်တဲ့အလေးအနက်ဆွေးနွေးမှုတွေဟာအမြဲတမ်း system administration ခေါင်းစဉ်တွေဆီကိုဦးတည်သွားတတ်ကြပေမယ့်ယခုစာအုပ်မှာတော့ system administration နဲ့ပက်သက်တဲ့ပြဿနာအနည်းငယ်ကိုပဲထိတွေ့ (ကိုင်တွယ်) သွားမှာဖြစ်ပါတယ် ။ မည်သို့ပင်ဆိုစေကာမူအဲ့ဒါကစာဖတ်သူကို အရေးကြီးတဲ့ system

administration task တွေမှာအသုံးပြုအတွက်မရှိမဖြစ် tool တခုဖြစ်တဲ့ command line အသုံးပြုမှုနဲ့ပတ်သက်ပြီး ထပ်ပေါင်းလေ့လာဖို့ ခိုင်မာတဲ့အခြေခံအုတ်မြစ်တွေပြင်ဆင်ဖြည့်ဆည်းပေးသွားမှာဖြစ်ပါတယ်။ ။

ယခုစာအုပ်ဟာ Linux ကိုအရမ်းဗဟိုပြုထားတဲ့စာအုပ်ဖြစ်ပါတယ်။ ။ အခြားစာအုပ်တွေကတော့လူကြိုက်များစေဖို့အတွက် Unix နဲ့ Mac OS X လိုအခြား platform တွေကိုထည့်ထားကြပါတယ်။ ။ အဲ့ဒီလိုလုပ်ခြင်းအားဖြင့်သူတို့စာအုပ်ထဲကပါဝင်တဲ့အချက်အလက်တွေကို ယျေဘုယျအနေနဲ့ပဲအပေါ်ယံလျှပ်ပြီးတော့ဖော်ပြသွားခဲ့ကြပါတယ်။ ။ အခြားတဖက်မှာတော့ ယခုစာအုပ်ဟာ ခေတ်ပြိုင် Linux distribution တွေကို cover ကာမိအောင်လုပ်ထားတဲ့အတွက် ၉၅ % သောစာအုပ်ထဲမှာပါဝင်တဲ့အချက်အလက်တွေဟာ အခြား Unix မျိုးတူစနစ် Unix-like system တွေအတွက်ပါအသုံးဝင်ပါတယ်။ ။ ဒါပေမယ့် ယခုစာအုပ်ဟာ ခေတ်သစ် Linux command line user တွေကိုသာအဓိကပစ်မှတ်ထား(ရည်ရွယ်)တာဖြစ်ပါတယ်။ ။

Who Should Read This Book

ယခုစာအုပ်ဟာ အခြား platform ကနေ ပြောင်းရွှေ့လာတဲ့ Linux user အသစ်တွေအတွက်ဖြစ်ပါတယ်။ ။ ဖြစ်နိုင်ခြေရှိတာကတော့ခင်ဗျားက Microsoft Windows ရဲ့ version အချို့ရဲ့ power user ဖြစ်ကောင်းဖြစ်နိုင်ပါတယ်။ ။ ခင်ဗျားရဲ့ boss က ခင်ဗျားကို Linux server တခုမှာ administration ပိုင်းကိုကိုင်ခိုင်းတာမျိုးဖြစ်နိုင်သလို ဒါမှမဟုတ် ခင်ဗျားဟာ security ပိုင်းဆိုင်ရာတွေနဲ့ပတ်သက်ပြီးစိတ်ကုန်လာပြီးတော့ Linux ကိုစမ်းသုံးကြည့်ချင်လာတဲ့ သာမန် desktop အသုံးပြုသူတစ်ယောက်လည်းဖြစ်နိုင်ပါတယ်။ ။ (ဘယ်လိုမျိုးပဲဖြစ်ပါစေ) အကုန်လုံးကိုဒီကနေကြိုဆိုပါတယ်။ ။

ပြောလေ့ရှိတာကတော့ Linux နဲ့ပတ်သက်ပြီး ဉာဏ်အလင်းပွင့်ဖို့ရာမှာ ဖြတ်လမ်း မရှိပါဘူး။ ။ command line ကိုသင်ယူရတာဟာတကယ့်စိမ်းခေါ်မှုတရပ်ဖြစ်ပြီးတော့တကယ့်ကိုကြိုးစားအားထုတ်ဖို့လိုအပ်ပါတယ်။ ။ အဲ့ဒါကအရမ်းခက်ခဲနေတာမျိုးမဟုတ်ပဲ အရမ်းများပြားနေတာမျိုးပါ။ ။ ပျမ်းမျှ Linux system တခုမှာတိုက်ရိုက်အားဖြင့်အသုံးပြုလို့ရတဲ့ program တွေ ထောင်နဲ့ချီပြီးပါဝင်ပါတယ်။ ။ မိမိကိုယ်ကိုသတိပေးလိုက်ပါ။ ။ command line ကိုလေ့လာတယ်ဆိုတာ သာမန်ကာလျှံကာကြိုးစားအားထုတ်မှုမျိုးမဟုတ်ပါဘူးဆိုတာကို။ ။

တဖက်မှာကလည်း Linux command line ကိုသင်ကြားရခြင်းကြောင့်ရရှိမယ့်ဆုလာဘ်တွေကလည်းအလွန်များပြားလှပါတယ်။ ။ ခင်ဗျားကိုခင်ဗျား power user တယောက်လို့ထင်နေရင် အခု ခဏနေပါဦး။ ။ တကယ့် power ဆိုတာဘာလဲဆိုတာကိုခင်ဗျား အခုအထိ တကယ်မသိသေးပါဘူး။ ။ ပြီးတော့အခြား computer အရည်အချင်းတွေ ၊ အသိပညာတွေနဲ့မတူတဲ့အချက်က command line က ရေရှည်ခံပါတယ်။ ။ ယနေ့သင်ကြားထားတဲ့အရည်အချင်းတွေက နောက် ၁၀ နှစ်လောက်အထိအသုံးဝင်နေမှာဖြစ်ပါတယ်။ ။ command line ဟာ အချိန် ရဲ့ စစ်ဆေးမှုကိုကျော်လွန်ရှင်သန်နိုင်ခဲ့ပါတယ်။ ။

ခင်ဗျားအနေနဲ့ programming အတွေ့အကြုံမရှိရင်လည်း စိတ်မပူပါနဲ့လို့ဆိုလိုပါတယ်။ ။ ကျွန်တော်တို့အနေနဲ့ခင်ဗျားကိုအဲ့ဒီလမ်းကြောင်းပေါ်ကိုလည်းအစက နေတင်ပေးမှာဖြစ်ပါတယ်။ ။

What's in This Book

ယခုစာအုပ်ထဲကအရာတွေကိုသေသေချာချာအစဉ်လိုက်ရွေးချယ်ပြီးတော့တင်ပြထားတာဖြစ်ပါတယ်။ ။ ဆရာတယောက်ကခင်ဗျားအနားမှာထိုင်ပြီးတော့ တောက်လျှောက်ခင်ဗျားကို Guide လုပ်ပေးနေသလိုမျိုးပေါ့။ ။

စာရေးသူအတော်များများက ဒီလိုအရာတွေကို systematic ပုံစံနဲ့ ကိုင်တွယ်လေ့ရှိကြတဲ့အခါ စာရေးသူတွေရှုထောင့်က နေ

ကြည့် ရင်အဓိပ္ပါယ်ရှိပေမယ့် အခုမှစတင်အသုံးပြုမယ့် new user တွေအတွက်တော့ အလွန်ရှုပ်ထွေးစရာဖြစ်သွားပါလိမ့်မယ်။

နောက်ထပ်ရည်မှန်းချက်ပန်းတိုင်တစ်ခုကတော့ခင်ဗျားကို Windows နည်းလမ်းအတိုင်းတွေးခေါ်ခြင်းနဲ့မတူတဲ့ Unix နည်းလမ်းအတိုင်းတွေးခေါ်တတ်ခြင်းနဲ့ရင်းနှီးကျွမ်းဝင်စေချင်လို့ပါ။ သင်ကြားနေတဲ့တလျှောက်မှာလည်း ကျွန်တော်တို့ကနေခင်ဗျားကို (သင်ကြားမှုအပြင်က) ခရီးတိုလေးတွေဆီကိုခေါ်ဆောင်သွားပြီးတော့ ဘာကြောင့်တချို့အရာတွေကအတိအကျအလုပ်လုပ်နေတာလဲဆိုတာရယ် ၊ ဘာလို့အဲ့လိုမျိုးဖြစ်နေရလဲဆိုတာရယ်ကို နားလည်အောင်အကူအညီပေးသွားမှာဖြစ်ပါတယ်။ Linux ဆိုတာ software အစိတ်အပိုင်းတစ်ခုဆိုတာထက် ကြီးမားတဲ့ Unix ယဉ်ကျေးမှုကြီး ရဲ့ သေးငယ်တဲ့အစိတ်အပိုင်းတစ်ခုဖြစ်ပြီးတော့သူ့မှာ ကိုယ်ပိုင်ဘာသာစကား နဲ့သမိုင်းကြောင်းရှိပါတယ်။ ကျွန်တော်လည်းပဲ(စိတ်လှုပ်ရှားပြီးတော့)တချက် နှစ်ချက်လောက်တော့ အသံကျယ်ကြီးနဲ့ အော်ဟစ်မိနိုင်ပါတယ်။ xD

ယခုစာအုပ်ကိုအပိုင်း ၄ ပိုင်းခွဲထားပြီးတော့တပိုင်းချင်းစီမှာ command line အတွေ့အကြုံတွေရဲ့ အမြင်ရှုထောင့်တွေကိုတင်ပြထားပါတယ်။

- **Part 1: Learning the Shell** ကိုစတင်သင်ယူခြင်းဖြင့် command line ရဲ့အခြေခံ language ကိုစူးစမ်းလေ့လာခြင်းဖြစ်တဲ့ command တွေရဲ့ structure တည်ဆောက်ပုံတွေ ၊ filesystem navigation တွေ ၊ command line မှာ editing လုပ်ပုံတွေ နဲ့ command ရဲ့ help တွေ document တွေရှာဖွေခြင်းတွေပါဝင်ပါတယ်။
- **Part 2: Configuration and the Environment** အပိုင်းမှာ computer ရဲ့ operation အပိုင်းတွေကို ထိန်းချုပ်ထားတဲ့ configuration file တွေကို command line ကနေ editing လုပ်ခြင်းတွေပါဝင်ပါတယ်။
- **Part 3: Common Tasks and Essential Tools** အပိုင်းမှာ command line ကနေအသုံးများတဲ့ ရိုးရိုးသာမန်လုပ်ဆောင်ချက် operation အပိုင်းတွေကိုစူးစမ်းလေ့လာကြမှာဖြစ်ပါတယ်။ Linux လိုမျိုး Unix-like operating system တွေမှာ data တွေအပေါ် powerful ဖြစ်တဲ့လုပ်ဆောင်ချက်တွေကိုလုပ်တဲ့အခါသုံးတဲ့ classic ရှေးကျတဲ့ command line program တွေပါဝင်ပါတယ်။
- **Part 4: Writing Shell Scripts** အပိုင်းမှာ shell programming ကိုမိတ်ဆက်ပေးမှာဖြစ်ပြီးတော့ အခြေခံလောက်ပဲဖြစ်နေတာကိုတော့ဝန်ခံရမှာပါ။ ဒါပေမယ့် လေ့လာရတာလွယ်ကူပါတယ်။ များစွာသော သာမန် computer လုပ်ဆောင်ချက်တွေအတွက် automating (program ကအလိုအလျောက်လုပ်ပေးခြင်း) technique နည်းလမ်းတွေကိုပေါ့။ shell programming ကိုလေ့လာသင်ယူခြင်းအားဖြင့် ခင်ဗျားအနေနဲ့ အခြား programming language ပေါင်းမြောက်များစွာမှာအသုံးပြုလို့ရတဲ့ programming နဲ့ပတ်သက်တဲ့ concept တွေနဲ့ရင်းနှီးလာမှာဖြစ်ပါတယ်။

How to Read This Book

အစ ကနေစဖတ်ပြီးတော့အဆုံးထိလိုက်ဖတ်သွားလိုက်ပါ။ ဒီစာအုပ်က reference work အနေနဲ့ရေးထားတာမဟုတ်ပါဘူး။ အစ ၊ အလယ် နဲ့ အဆုံး ပါဝင်တဲ့ဇာတ်လမ်းတပုဒ်လိုပါ။

Prerequisites

ယခုစာအုပ်ကိုအသုံးပြုဖို့အတွက်ခင်ဗျားအနေနဲ့လိုအပ်တာကတော့အလုပ်လုပ်လို့ရတဲ့ working linux installation တခုပါပဲ။ အဲဒါကိုခင်ဗျားအနေနဲ့အောက်ပါနည်းလမ်း ၂ ခုထဲက တခုနဲ့ရယူနိုင်ပါတယ်။

- **Install Linux on a (not so new) computer.** ခင်ဗျားအနေနဲ့ဘယ်လို distribution ကိုရွေးချယ်တယ်ဆိုတာအရေးကြီးပါဘူး ဆိုပေမယ့်လည်းယနေ့ခေတ်လူအတော်များများကတော့ Ubuntu ၊ Fedora သို့မဟုတ် OpenSUSE နဲ့စတင်လေ့လာနေကြပါတယ်။ တကယ်လို့ဘာသုံးရမယ်ဆိုတာမသေချာခဲ့ရင် Ubuntu ကိုအရင်စမ်းကြည့်ပါ။ ခေတ်သစ် Linux distribution တွေကို installing သွင်းခြင်းကခင်ဗျားရဲ့ hardware အပေါ်မူတည်ပြီး အံ့ဩစရာကောင်းလောက်အောင်ကိုလွယ်ကူနေတာ ဒါမှမဟုတ် အံ့ဩစရာကောင်းလောက်အောင်ကိုခက်ခဲနေပါလိမ့်မယ်။ ကျွန်တော်အကြံပေးလိုတာကတော့ နှစ်အတန်ကြာအိုနေပြီဖြစ်တဲ့ desktop တလုံးမှာ RAM က 256 MB နဲ့ Hard disk space က 60 Gb ရှိရင်ရပါပြီ။ laptop တွေနဲ့ wireless network တွေက လုပ်ရကိုင်ရတာကိုခက်ခဲစေတာမို့သူတို့ကိုရှောင်ပါ။ (အခုခေတ်မှာရှောင်စရာမလိုတော့ပါ။ ရဲရဲသာ install လုပ်ပါ။)
- **Use a live CD.** Linux distribution တွေအများကြီးပေါ်မှာ ခင်ဗျားလုပ်နိုင်တဲ့ အရမ်းမိုက်တဲ့ ကိစ္စတွေအများကြီးထဲကတခုကတော့ သူတို့တွေကိုသွင်းစရာမလိုပဲ CD-ROM (သို့မဟုတ် Usb , external hdd, memory card) ကနေ တိုက်ရိုက် run လို့ရတာပဲဖြစ်ပါတယ်။ ခင်ဗျား (computer) ရဲ့ BIOS setup ထဲသွားပြီးတော့ computer ကို Boot from (CD-ROM , Usb , external hdd, memory card တို့ထဲကတခုခု) ကိုပေးပြီး live CD ထည့်ပြီးတော့ reboot ချလိုက်ပါ။ live CD ကိုအသုံးပြုခြင်းဟာ Linux installation လုပ်ဖို့အတွက်အရေးကြီးတဲ့ Linux compatibility ရှိမရှိခင်ဗျားရဲ့ computer ကိုစမ်းဖို့အတွက်အရမ်းကောင်းတဲ့နည်းလမ်းတခုပါပဲ။ live CD တချပ်ကိုသုံးတဲ့အခါ အားနည်းချက်ကတော့ Linux ကို hard drive ထဲ install သွင်းပြီးသုံးတာထက်စာရင် အရမ်းကို နှေးကွေးနေတာပဲဖြစ်ပါတယ်။ Ubuntu နဲ့ Fedora နှစ်ခုစလုံးမှာ (အခြားဟာတွေမှာရော) live CD version တွေပါဝင်ကြပါတယ်။

(တကယ်တော့အခုခေတ်မှာ linux installation သွင်းတာမှာ နည်းအများကြီးရှိနေပါပြီ။ အဆင်ပြေရာနည်းလမ်းတချို့ကိုမိမိဖာသာ youtube ကနေကြည့်ပြီး install လုပ်နိုင်ပါတယ်။ Ubuntu linux သွင်းနည်းတချို့ကို youtube ကနေကြည့်နိုင်အောင် link တချို့ထည့်ပေးလိုက်ပါတယ်။

Ubuntu 20.04 LTS Linux Install Tutorial | 2021 Desktop Version | (Linux Beginners) - Focal Fossa

<https://www.youtube.com/watch?v=CFI1Jt8kVUk>

Ubuntu Complete Beginner's Guide: Download & Installing Ubuntu

<https://www.youtube.com/watch?v=W-RFY4LQ6oE>

How To Install Ubuntu And Keep Windows: Dual Boot Tutorial

<https://www.youtube.com/watch?v=x1ykDpSzpKU>

How to Install Ubuntu 20.04 LTS on VirtualBox in Windows 10

<https://www.youtube.com/watch?v=x5MhydijWmc>

How to Make Ubuntu 20.04 Bootable USB Drive

https://www.youtube.com/watch?v=X_fDdUgqIUQ)

Note : ခင်ဗျားအနေနဲ့ linux ကိုဘယ်လိုပဲ install လုပ်သည်ဖြစ်စေ ခင်ဗျားအနေနဲ့ယခုစာအုပ်
ထဲကသင်ခန်းစာတွေကိုလိုက်လုပ်ဖို့အတွက် superuser (administrative) privilege တွေကိုရုံဖန်ရုံခါ
လိုအပ်မှာဖြစ်ပါတယ်။

ခင်ဗျားအနေနဲ့အလုပ်လုပ်လို့ရတဲ့ working installation သွင်းပြီးပြီဆိုတာနဲ့ စတင် (စာအုပ်ကို) ဖတ်ပြီး
တော့ခင်ဗျားရဲ့ကိုယ်ပိုင် computer နဲ့လိုက်လုပ်ကြည့်ပါ။ ။ ယခုစာအုပ်ထဲကအချက်အလက်တွေအများစုက
လက်တင်သုံးရုံပဲဆိုတော့ ထိုင်ပြီးတော့ Typing ရိုက်ပေတော့။

WHY I DON'T CALL IT "GNU/LINUX"

တချို့လူတွေကြားမှာ Linux operating system ကို ဥပဒေအရ “GNU/Linux operating system.” လို့
ပြင်ဆင်ခေါ်ဆိုကြပါတယ်။ ။ “Linux” နဲ့ပတ်သက်တဲ့ ပြဿနာကတော့ သူ့ကိုပြီးပြည့်စုံမှန်ကန်နေအောင်နာမည်ပေး
လို့မရပါဘူး။ ။ ဘာလို့လဲဆိုတော့ သူ့ကိုဖန်တီးထုတ်ဝေဖြန့်ချိဖို့အတွက်ကြိုးစားရာမှာ မတူညီတဲ့လူပေါင်းမြောက်များ
စွာကဝိုင်းရေးထားလို့ပါ။ ။ နည်းပညာအရပြောရမယ်ဆိုရင် Linux ဆိုတာ operating system ရဲ့ kernel နာမည်
သာဖြစ်ပြီးတော့ဒီထက်မပိုပါဘူး။ ။ kernel ကအရမ်းအရေးကြီးပါတယ်။ ။ ဟုတ်ပါတယ်။ ။ သူက operating system
ကိုမောင်းနှင်နိုင်တယ်ဆိုကထဲကပေါ့။ ။ ဒါပေမယ့် ပြီးပြည့်စုံတဲ့ operating system ကြီးတခုအဖြစ်ပုံဖော်နိုင်လောက်
အောင်တော့မဟုတ်ပါဘူး။

Richard Stallman အကြောင်းပါလာပါပြီ။ ။ Free software လှုပ်ရှားမှုကိုစတင်ခဲ့တဲ့ ပါရမီရှင်
အတွေးအခေါ်ပညာရှင် ၊ Free Software Foundation ကိုစတင်ခဲ့တယ် ၊ GNU project ကိုပုံစံချပေးခဲ့တယ် ၊ ပထမ
ဆုံးသော GNU C Compiler (GCC) ကိုရေးသားခဲ့တယ် ၊ GNU General Public License (the GPL) ကိုဖန်တီးခဲ့
တယ် စသည်ဖြင့် စသည်ဖြင့်။ ။ သူကခင်ဗျား(တို့)ကို GNU Project က အထောက်အပံ့ပေးခဲ့တာကိုမှန်မှန်ကန်ကန်
ထင်ဟပ်နိုင်စေဖို့အတွက် “GNU/Linux” လို့ခေါ်ဖို့တိုက်တွန်းခဲ့တယ်။ ။ GNU Project က Linux kernel ထက်အရင်
ဦးပြီးတော့ Project ရဲ့အထောက်အပံ့ပေးမှုတွေကအလွန်အင်မတန်မှ အသိအမှတ်ပြုပေးဖို့ထိုက်တန်တဲ့အခါမှာ သူ
တို့ကို နာမည်ထဲမှာထည့်ထားတာဟာ အခြား အထောက်အပံ့ပေးသူတွေအပေါ်မှာ မတရားလုပ်သလိုဖြစ်နေပါတယ်
။ နောက်တချက်ကလည်း kernel ကအရင် boot တက်ပြီးမှကျန်တဲ့အခြားဟာတွေကသူ့အပေါ်မှာ run ရတယ်ဆိုက
ထဲက “Linux/GNU” လို့ခေါ်တာက နည်းပညာအရပိုပြီးတိကျမယ်လို့ကျွန်တော်ထင်မိတယ်။

လူသုံးများတဲ့ (လူတွေအခေါ်များတဲ့) “Linux” ဆိုတာ kernel နဲ့ ပုံမှန် Linux distribution တွေမှာ တွေ့ရတတ်တဲ့ free and open source software တွေကိုရည်ညွှန်းပါတယ်။ ဒါပေမယ့် Linux ecosystem ကြီးတစ်ခုလုံးကိုခေါ်တာပါ။ GNU component တွေသက်သက်ကိုခေါ်တာမဟုတ်ပါဘူး။ operating system ဈေးကွက်ထဲမှာတော့ DOS, Windows, Solaris, Irix, AIX တို့လိုနာမည်တလုံးထဲပါတာကိုပိုပြီးကြိုက်ကြပုံရပါတယ်။ ကျွန်တော့်အနေနဲ့ကတော့ ကျော်ကြားတဲ့ပုံစံကိုသုံးရတာပိုကြိုက်ပါတယ်။ တကယ်လို့ ဘယ်လိုပဲဖြစ်ဖြစ် ခင်ဗျားအနေနဲ့ “GNU/Linux” လို့ပဲသုံးမယ်ဆိုရင် ဒီစာအုပ်ကိုဖတ်နေတဲ့အချိန်မှာတော့ ကျေးဇူးပြုပြီးတော့ စိတ်ပိုင်းဆိုင်ရာ ပြောင်းလဲမှုပြုလုပ်ပြီးတော့အစားထိုးလိုက်ပါ။ ကျွန်တော်စိတ်မဆိုးပါဘူး။

PART 1

LEARNING THE SHELL

1

WHAT IS THE SHELL?

ကျွန်တော်တို့အနေနဲ့ command line အကြောင်းပြောတဲ့အခါမှာကျွန်တော်တို့တကယ်တမ်းရည်ညွှန်းပြောဆိုနေတာက shell ဖြစ်ပါတယ်။ shell ဆိုတာက keyboard က command တွေကိုယူဆောင်ပြီးတော့ operating system ဆီကိုပို့ပေးတဲ့ program တခုဖြစ်ပါတယ်။ Linux distribution တွေအကုန်လုံးနီးပါးမှာ GNU Project က bash လို့ခေါ်တဲ့ shell program တခုထည့်သွင်းပေးထားလေ့ရှိပါတယ်။ bash ဆိုတဲ့နာမည်က Bourne Again Shell ရဲ့အတိုကောက်နာမည်ဖြစ်ပါတယ်။ အမှန်အတိုင်းအချက်အလက်နဲ့ပြောရမယ်ဆိုရင်တော့ bash ဆိုတာ sh ကိုပိုကောင်းအောင်အစားထိုးထားတာပါ။ Steve Bourne ရေးသားခဲ့တဲ့ original Unix shell program တခုဖြစ်ပါတယ်။

Terminal Emulators

graphical user interface တခုကိုသုံးတဲ့အခါမှာ ကျွန်တော်တို့အနေနဲ့ shell နဲ့ interact လုပ်ဖို့အတွက် terminal emulator လို့ခေါ်တဲ့နောက်ထပ် program တခုလိုအပ်ပါတယ်။ ကျွန်တော်တို့တွေရဲ့ desktop menu တွေထဲမှာလိုက်ရှာကြည့်လိုက်ရင် တခုလောက်တော့တွေ့ကောင်းတွေ့နိုင်ပါတယ်။ KDE က konsole ကိုသုံးပြီးတော့ GNOME က gnome-terminal ကိုသုံးထားပေမယ့်လည်းကျွန်တော်တို့ရဲ့ menu ပေါ်မှာတော့ ရိုးရိုးလေး terminal လို့ပဲခေါ်ကောင်းခေါ်နိုင်ပါတယ်။ Linux အတွက် terminal emulator တွေအများအပြားရှိပေမယ့်လည်း သူတို့အားလုံးကအခြေခံအားဖြင့်တူညီတဲ့အလုပ်ကိုပဲလုပ်ပေးကြပါတယ်။ ကျွန်တော်တို့ကို shell ဆီကို access လုပ်ပေးတာပါ။ ခင်ဗျားအနေနဲ့သူတို့မှာရှိတဲ့ ထပ်ပေါင်းထည့်ထားတဲ့ ဒါမှမဟုတ် ထုတ်ပစ်လိုက်တဲ့ feature တွေအပေါ် အခြေခံပြီးတော့ တခု သို့မဟုတ် အခြားတခုကို တဖြည်းဖြည်းနှစ်သက်လာတာမျိုးဖြစ်ကောင်းဖြစ်နိုင်ပါတယ်။

Your First Keystrokes

ဒါဆို စကြစို့။ Terminal emulator ကိုဖွင့်လိုက်ပါ။ (သူ့ shortcut က Ctrl+Alt+T ပါ) သူတက်လာပြီဆို တာနဲ့ ခင်ဗျားအနေနဲ့ဒါမျိုးတခုခုမြင်တွေ့ရမှာဖြစ်ပါတယ်။

```
jok3r@lucy:~$
```

ဒါကို shell prompt လို့ခေါ်ပါတယ်။ ပြီးတော့ shell က input ကိုလက်ခံဖို့အတွက် ready ဖြစ်နေတဲ့ အချိန်တိုင်းမှာသူကပေါ်လာပါတယ်။ distribution အပေါ်မူတည်ပြီးတော့သူပုံစံကအမျိုးမျိုးကွဲပြားနေနိုင်ပေမယ့် ပုံမှန်အားဖြင့်တော့ `username@machinename` ကိုထည့်ထားပေးပြီးတော့ သူ့နောက်မှာ current working directory (ခဏကြာရင်ဒီအကြောင်းကိုပိုစုံအောင်ပြောမှာပါ) နဲ့ dollar sign (\$) ပါလာပါတယ်။

တကယ်လို့ prompt ရဲ့နောက်ဆုံး character က dollar sign (\$) မဟုတ်ဘဲ hash mark (#) ဖြစ်နေခဲ့မယ်ဆိုရင် terminal session မှာ superuser privilege ရှိနေပါပြီ။ ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ ကျွန်တော်တို့က root user အနေနဲ့ log in ဝင်ထားတာ ဒါမှမဟုတ် ကျွန်တော်တို့က superuser (administrative) privilege ပေးနိုင်တဲ့ terminal emulator တခုကိုရွေးချယ်ထားတာ စတဲ့နှစ်မျိုးထဲကတမျိုးဖြစ်နိုင်ပါတယ်။

အခုအထိတော့အခြေအနေကောင်းနေတယ်လို့မှတ်ယူလိုက်ပြီးတော့ typing တချို့ရိုက်ကြည့်ကြစို့။ prompt ထဲမှာအဓိပ္ပါယ်မရှိတဲ့စာတွေလျှောက်ရိုက်ပြီး (Enter ခေါက်) ကြည့်လိုက်ပါ။

```
jok3r@lucy:~$ kaekfjaeifj
```

ဒီ command ကဘာအဓိပ္ပါယ်မှမရှိတဲ့အတွက် shell ကလည်းဘာအဓိပ္ပါယ်မှမရှိကြောင်းပြောပြပြီးတော့ ကျွန်တော်တို့ကို (command ထပ်ရိုက်ဖို့အတွက်) နောက်ထပ်အခွင့်အရေးပေးလိုက်ပါတယ်။

```
bash: kaekfjaeifj: command not found
```

```
jok3r@lucy:~$
```

Command History

ကျွန်တော်တို့တွေ (keyboard ပေါ်က) up-arrow key ကိုတချက်နှိပ်လိုက်မယ်ဆိုရင် အရင်ကရိုက်ခဲ့တဲ့ command ဖြစ်တဲ့ kaekfjaeifj က prompt အနောက်မှာပြန်ပေါ်လာမှာဖြစ်ပါတယ်။ အဲ့ဒါကို command history လို့ခေါ်ပါတယ်။ Linux distribution အတော်များများမှာ default အနေနဲ့ နောက်ဆုံး command 500 လောက်ကို မှတ်ထားပေးနိုင်ပါတယ်။ (default အနေနဲ့မှတ်ပေးတာ 500 ပါ။ အတိုးအလျော့လုပ်လို့ရပါသေးတယ်) down-

arrow ကိုတချက်နှိပ်လိုက်မယ်ဆိုရင် အရင်ကရိုက်ခဲ့တဲ့ command ဖြစ်တဲ့ kaekfjaeifj ပြန်ပျောက်သွားမှာဖြစ်ပါတယ်။ ။

Cursor Movement

အရင်ကရိုက်ခဲ့တဲ့ command ကို up-arrow key ကိုတချက်နှိပ်လိုက်ပြီးတော့ပြန်ခေါ်လိုက်ပါ။ ။ အခု left နဲ့ right arrow key တွေကိုစမ်းကြည့်လိုက်ပါ။ ။ ကျွန်တော်တို့အနေနဲ့ command line ပေါ်ကဘယ်နေရာမှာမဆို cursor ကိုနေရာချလို့ရတာကိုတွေ့ရပါလိမ့်မယ်။ ။ အဲဒါက command တွေကို editing လုပ်ရာမှာလွယ်ကူစေပါတယ်။ ။

A FEW WORDS ABOUT MICE AND FOCUS

shell ဆိုတာ keyboard နဲ့ပက်သက်တဲ့အကြောင်းချည်းပဲလို့ပြောနေစဉ်မှာပဲ ခင်ဗျားအနေနဲ့ ခင်ဗျားရဲ့ Terminal emulator ပေါ်မှာ mouse တခုနဲ့လည်းအသုံးပြုနိုင်ပါတယ်။ ။ X Window System (GUI အလုပ်လုပ်စေဖို့ သူ့အောက်မှာထည့်သွင်းထားတဲ့အင်ဂျင်) အထဲကိုတည်ဆောက်ထည့်သွင်းထားတဲ့ mechanism မှာ လျှင်မြန်တဲ့ copy-and-paste နည်းပညာတခုကို support ပေးပါတယ်။ ။ တကယ်လို့ခင်ဗျားအနေနဲ့စာတချို့ကို highlight လုပ်ဖို့အတွက် left mouse button ကိုဖိထားပြီးတော့ mouse ကိုရွှေ့လိုက်မယ်ဆိုရင် (သို့မဟုတ် double click လုပ်လိုက်မယ်ဆိုရင်) အဲဒီ (highlight လုပ်ထားတဲ့စာတွေ) ဟာ တွေက X ကနေ maintain လုပ်ထားတဲ့ buffer ထဲကို copy လုပ်သွားမှာဖြစ်ပါတယ်။ ။ middle mouse button ကိုနှိပ်လိုက်ရင်တော့ cursor ချထားတဲ့နေရာမှာ စာတွေကို paste သွားချပေးမှာဖြစ်ပါတယ်။ ။ စမ်းကြည့်လိုက်ပါ။ ။ (တကယ်လို့ခင်ဗျားက highlight လုပ်တဲ့အချိန်မှာ prompt အနောက်က `username@localhost` ထဲကစာတွေကိုပါ highlight လုပ်မိသွားရင် paste ချတဲ့အခါမှာ အဲဒီစာတွေပါ ပါလာမှာဖြစ်ပြီးတော့ error တက်နိုင်ပါတယ်။)

terminal ထဲမှာ copy and paste လုပ်ဖို့အတွက် CTRL -C နဲ့ CTRL -V ကိုသုံးဖို့မကြိုးစားပါနဲ့။ ။ အလုပ်လုပ်မှာမဟုတ်ပါဘူး။ ။ (နောက်ပိုင်းထွက်တဲ့ Ubuntu terminal ထဲမှာတော့ CTRL+Shift +C နဲ့ CTRL+Shift +V ဆိုရင် copy and paste လုပ်လို့ရပါတယ်။ ။ middle mouse wheel ကိုနှိပ်ရင် paste လုပ်ပေးပါတယ်။) shell အတွက်တော့ အဲဒီ control code တွေဟာ Microsoft Windows မပေါ်ခင်ကနှစ်ပေါင်းများစွာကထဲက မတူညီတဲ့တခြားအဓိပ္ပါယ်တခု (တခြားအလုပ်တခုကိုလုပ်ပေးဖို့) သတ်မှတ်ထားပြီးသားဖြစ်နေလို့ပါပဲ။ ။

ခင်ဗျားရဲ့ graphical desktop environment (KDE သို့မဟုတ် GNOME ဖြစ်ဖို့များပါတယ်) မှာ Windows လိုပြုမူလုပ်ဆောင်စေဖို့အတွက် အားထုတ်ထားတဲ့အထဲမှာ focus policy ကို “Click to focus” မှာ set လုပ်ထားကောင်းထားနိုင်ပါတယ်။ ။ ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ window တခုကို focus လုပ်ခိုင်းထားဖို့အတွက် (active ဖြစ်နေဖို့အတွက်) ဆိုရင် ခင်ဗျားအနေနဲ့ (focus ဖြစ်ချင်တဲ့ windows ကို) click လုပ်ဖို့လိုအပ်ပါတယ်။ ။ အဲဒါက ရိုးရာ X ရဲ့ “focus follows mouse” နဲ့တော့ဆန့်ကျင်ဖက်ဖြစ်နေပါတယ်။ ။ ဆိုလိုတာက mouse က window အပေါ် ဖြတ်သွားတာနဲ့ window က active ဖြစ်သွားမယ်လို့ပြောတာပါ။ ။ ခင်ဗျား click မလုပ်မချင်း window က foreground ကိုရောက်လာမှာမဟုတ်ပါဘူး။ ။ ဒါပေမယ့် သူက input တွေကိုတော့လက်ခံပေးမှာပါ။ ။ focus policy ကို “focus follows mouse” မှာထားခြင်းအားဖြင့် terminal window တွေကိုအသုံးပြုရာမှာပိုမိုလွယ်ကူစေမှာဖြစ်ပါတယ်။ ။ စမ်းကြည့်လိုက်ပါ။ ။ ခင်ဗျားတခါလောက်စမ်းကြည့်လိုက်ရင်ကြိုက်သွားမယ်လို့ကျွန်တော်ထင်ပါတယ်။ ။

ခင်ဗျားအနေနဲ့ အဲဒီ setting ကို ခင်ဗျားရဲ့ window manager ထဲက configuration program ထဲမှာရှာတွေ့နိုင်ပါတယ် ။ (Ubuntu 20.04 မှာဆိုရင်တော့ “Click to focus” က default ပါ ။)

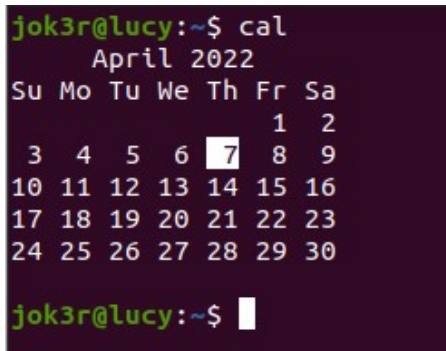
Try Some Simple Commands

ကျွန်တော်တို့အခု စာရိုက်တတ်သွားပြီဆိုတော့ (terminal ထဲမှာပြောတာပါ xD) ရိုးရင်းတဲ့ command တချို့ကို စမ်းကြည့်ကြစို့ ။ ပထမဆုံးတခုကတော့ date ဖြစ်ပါတယ် ။ ဒီ command က လက်ရှိ အချိန် နဲ့ ရက်စွဲကိုပြပေးပါတယ် ။

```
jok3r@lucy:~$ date
```

```
Thu 07 Apr 2022 10:37:05 PM +0630
```

သူနဲ့ဆက်နွှယ်တဲ့ command ကတော့ cal ဖြစ်ပါတယ် ။ သူက default အရ လက်ရှိ လ ရဲ့ ပြက္ခဒိန် ကိုပြပေးပါတယ် ။



```
jok3r@lucy:~$ cal
April 2022
Su Mo Tu We Th Fr Sa
    1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
jok3r@lucy:~$
```

ခင်ဗျားရဲ့ disk drive (harddisk) ထဲက လက်ရှိ free space ဘယ်လောက်ရှိသလဲကြည့်ချင်ရင် df လို့ရိုက်ထည့်လိုက်ပါ ။

```
jok3r@lucy:~$ df
```

Filesystem	1K-blocks	Used	Available	Use%	Mounted on
udev	3978492	0	3978492	0%	/dev
tmpfs	802744	2160	800584	1%	/run
/dev/sda5	170980648	135963340	26262264	84%	/
tmpfs	4013704	0	4013704	0%	/dev/shm
tmpfs	5120	4	5116	1%	/run/lock

tmpfs	4013704	0	4013704	0%	/sys/fs/cgroup
/dev/loop0	190976	190976	0	100%	/snap/arduino/56

ထို့အတူပါပဲ free memory ဘယ်လောက်ကျန်သလဲပြဖို့အတွက် free ဆိုတဲ့ command ကိုရိုက်ထည့်လိုက်ပါ။

```
jok3r@lucy:~$ free
```

	total	used	free	shared	buff/cache	available
Mem:	8027408	2054660	2884580	370444	3088168	5335152
Swap:	2097148	0	2097148			

Ending a Terminal Session

ကျွန်တော်တို့အနေနဲ့ terminal session တခုကို terminal emulator window ကိုပိတ်ခြင်း သို့မဟုတ် shell prompt မှာ exit ဆိုတဲ့ command ကိုရိုက်ထည့်ခြင်းစတဲ့နှစ်နည်းထဲကတခုနည်းနည်းနဲ့ပိတ်လို့ရပါတယ်။

```
jok3r@lucy:~$ exit
```

THE CONSOLE BEHIND THE CURTAIN

တကယ်လို့ကျွန်တော်တို့မှာ terminal emulator running ဖြစ်နေရင်တောင်မှ graphical desktop (GUI) ရဲ့နောက်ကွယ်မှာ terminal emulator session တွေအများအပြား run နေကြပါတယ်။ virtual terminal တွေ သို့မဟုတ် virtual console တွေလို့ခေါ်တဲ့အဲ့ဒီ session တွေကို Linux distribution အတော်များများမှာ CTRL - ALT - F1 ကနေ CTRL - ALT - F6 အထိ (တခုမဟုတ်တခုကိုနှိပ်ပြီး) access လုပ်နိုင်ပါတယ်။ session တခုကို access လုပ်လိုက်တာနဲ့ username နဲ့ password ရိုက်ထည့်လို့ရတဲ့ log in prompt ပေါ်လာပါတယ်။ virtual console တခုကနေ နောက်တခုကိုပြောင်းပြီးသွားဖို့အတွက် ALT နဲ့ F1 ကနေ F6 အထိနှစ်သက်ရာနှိပ်လို့ရပါတယ်။ graphical desktop ဆီကိုပြန်သွားချင်တယ်ဆိုရင်တော့ ALT + F7 ကိုနှိပ်လိုက်ပါ။

(terminal ထဲမှာကိုယ်ရိုက်ထားတဲ့ command တွေများလာလို့မျက်စိနောက်ပြီး screen ကိုရှင်းပစ်လိုက်ချင်ရင် clear command ကိုသုံးရင်သုံး ဒါမှမဟုတ် Ctrl + l ကိုနှိပ်လိုက်ပါ။)

2

Navigation

ပထမဦးဆုံးကျွန်တော်တို့လေ့လာဖို့လိုအပ်တာကတော့ (typing ပြီးရင်) ကျွန်တော်တို့ရဲ့ Linux system ပေါ်က file system မှာဘယ်လို navigate လုပ်ရမလဲဆိုတာပါပဲ ။ အခု chapter မှာကျွန်တော်တို့အနေနဲ့အောက်ပါ command တွေကို မိတ်ဆက်ပေးသွားမှာဖြစ်ပါတယ် ။

- **pwd** — Print name of current working directory. (လက်ရှိအလုပ်လုပ်နေတဲ့ directory ရဲ့နာမည်ကို ထုတ်ပြပါမယ် ။)
- **cd** — Change directory. (directory နောက်တခုကိုကူးပြောင်းပေးပါမယ် ။)
- **ls** — List directory contents. (directory ထဲမှာရှိနေတဲ့ content တွေကို list ထုတ်ပြပါမယ် ။)

Understanding the Filesystem Tree

Windows တွေမှာလိုပဲ Unix-like operating system တခုဖြစ်တဲ့ Linux ဟာလည်းသူ့ရဲ့ file တွေကို hierarchical directory structure လို့ခေါ်တဲ့အရာတခုနဲ့ စုစည်းထားပါတယ် ။ ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ သူတို့ဟာ directory (တခြား system တွေမှာ folder လို့ခေါ်ကြပါတယ်) တွေကို သစ်ပင် pattern ပုံစံမျိုးနဲ့စုစည်းထားပြီးတော့သူ့ထဲမှာ file တွေနဲ့ အခြား directory တွေပါဝင်ကောင်းပါဝင်နိုင်ပါသေးတယ် ။ filesystem ထဲက ပထမဦးဆုံးသော directory ကို root directory လို့ခေါ်ပါတယ် ။ root directory ထဲမှာ file တွေနဲ့ အခြား subdirectory တွေပါဝင်ပြီးတော့ သူတို့ထဲမှာ (subdirectory တွေထဲမှာ)မှ နောက်ထပ် file တွေနဲ့ အခြား subdirectory တွေထပ်ပြီး အဆင့်ဆင့် ပါဝင်နေကြပါတယ် ။

တခုမှတ်ထားရမှာက windows နဲ့မတူတဲ့အချက်က သူ့မှာက storage device တခုစီတိုင်းအတွက် filesystem tree တခုစီသီးခြားရှိနေတာကိုပါပဲ ။ computer မှာ storage device ဘယ်နှစ်ခုတပ်ဆင်ထားသည်ဖြစ်စေ Linux လို Unix-like system တွေမှာ single filesystem tree တခုအမြဲတမ်းပါရှိပါတယ် ။ Storage device တွေဟာ system ကို maintenance လုပ်ဖို့အတွက်တာဝန်ရှိတဲ့ လူ (သို့မဟုတ် လူများ) ဖြစ်တဲ့ system administrator တွေရဲ့ စိတ်ကူးအတိုင်း tree ပေါ်ကများပြားလှတဲ့ point တွေမှာ attached (သို့မဟုတ် ပိုပြီးမှန်အောင်ပြောရမယ်ဆိုရင် mounted) လုပ်ထားပါတယ် ။

The Current Working Directory

ကျွန်တော်တို့တတွေတော်တော်များများက figure 2-1 မှာပြထားတဲ့အတိုင်း filesystem tree ကို ကိုယ်စားပြုတဲ့ graphical file manager နဲ့ရင်းနှီးကြပါတယ်။ သတိပြုရမှာက ပုံမှန်အားဖြင့် tree က အပေါ်မှာလမ်းဆုံးတဲ့ upended ဖြစ်ပါတယ်။ ဒါပေမယ့် root က အပေါ်ဆုံးမှာရှိနေပြီးတော့ အကိုင်းအခက်တွေအများကြီးက အောက်ကိုဆင်းသွားပါတယ်။

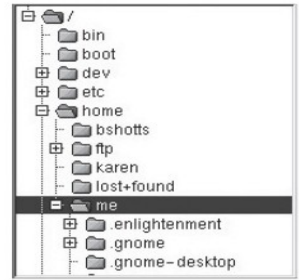


Figure 2-1: Filesystem tree as shown by a graphical file manager

ဘယ်လိုပဲဖြစ်ဖြစ် command line မှာ ပုံမပါပါဘူး။ filesystem tree မှာ လည်းထို့အတူပါပဲ။ ကျွန်တော်တို့အနေနဲ့တခြားနည်းလမ်းတခုနဲ့စဉ်းစားဖို့လိုအပ်လာပါတယ်။

filesystem ဆိုတာကို သစ်ပင်တပင်ကိုဇောက်ထိုးလုပ်ထားတဲ့ဝက်ဘာပုံစံတခုဖြစ် ပြီးတော့ကျွန်တော်တို့ကအဲ့ဒီအလယ်မှာရပ်နေနိုင်တယ်လို့ စိတ်ကူးယဉ်ကြည့်ပါ။ ဘယ်အချိန်မှာမဆို ကျွန်တော်တို့က directory တခုအတွင်းမှာရှိနေပြီးတော့ ကျွန်တော်တို့က directory ထဲမှာရှိနေတဲ့ file တွေ နဲ့ ကျွန်တော်တို့(လက်ရှိရောက်နေတဲ့ directory) ရဲ့အပေါ်က directory (parent directory လို့ခေါ်ပါတယ်) ကိုသွားတဲ့ pathway နဲ့ ကျွန်တော်တို့အောက်မှာရှိနေတဲ့ဘယ်လို subdirectory တွေကိုမဆိုမြင်နိုင်ပါတယ်။ ကျွန်တော်တို့လက်ရှိရောက်နေတဲ့ directory ကို current working directory လို့ခေါ်ပါတယ်။ current working directory ကိုဖော်ပြဖို့အတွက် ကျွန်တော်တို့အနေနဲ့ pwd (print working directory) command ကိုအသုံးပြုပါတယ်။

```
jok3r@lucy:~$ pwd
/home/jok3r
```

ကျွန်တော်တို့ ပထမဦးဆုံးအကြိမ် system ထဲကို log in ဝင်တဲ့အခါ (ဒါမှမဟုတ် terminal emulator session တခုကိုစတင်တဲ့အခါ) မှာ ကျွန်တော်တို့ရဲ့ current working directory ဟာ home directory မှာရောက်နေမှာ ဖြစ်ပါတယ်။ user account တခုစီတိုင်းကိုသူတို့ကိုယ်ပိုင် home directory ပေးထားပြီးတော့ အဲ့ဒါကလည်း user က regular user အနေနဲ့လုပ်ဆောင်တဲ့အခါမှာ file တွေကို write ရေးသားခွင့်ပြုတဲ့ တခုထဲသောနေရာလည်းဖြစ်ပါတယ်။

Listing the Contents of a Directory

current working directory အတွင်းက file တွေနဲ့ directory တွေကို list လုပ်ချင်တဲ့အခါမှာကျွန်တော်တို့က ls command ကိုအသုံးပြုပါတယ်။

```
jok3r@lucy:~$ ls
```

default.ovpn	dwhelper	mm-kb-master	Recordings
Desktop	hacking_tutorial_link	Music	snap
Documents	javasharedresources	Pictures	Templates
Downloads	master.zip	pt	Videos
Dropbox	MEGAsync	Public	

တကယ်တော့ ကျွန်တော်တို့အနေနဲ့ ls command ကို ဘယ် directory ထဲက content တွေကိုမဆို list ထုတ်ကြည့်တဲ့နေရာမှာအသုံးပြုနိုင်ပါတယ်။ current working directory ကိုပဲ list ထုတ်ကြည့်လို့ရတာမဟုတ်ပါဘူး။ ပြီးတော့အဲ့ဒါကတခြားပျော်စရာကောင်းတဲ့အရာတွေအများကြီးကိုလည်းပဲလုပ်ပေးနိုင်ပါသေးတယ်။ ကျွန်တော်တို့ Chapter 3 ရောက်တဲ့အခါမှာ ls command အတွက်အချိန်ပိုပေးပါမယ်။

Changing the Current Working Directory

ခင်ဗျားရဲ့ working directory ကိုပြောင်းဖို့အတွက် (ကျွန်တော်တို့ရပ်နေတဲ့သစ်ပင်ပုံစံဝက်ဘာထဲမှာ) ကျွန်တော်တို့က cd command ကိုအသုံးပြုပါတယ်။ cd ရဲ့အနောက်မှာခင်ဗျားသွားလိုတဲ့ working directory ရဲ့ pathname ကိုရိုက်ထည့်လိုက်ပါ။ pathname တခုဆိုတာကျွန်တော်တို့လိုချင်တဲ့ (သွားချင်တဲ့) directory ဆီကိုရောက်ဖို့အတွက် tree သစ်ပင်ရဲ့အကိုင်းအခက်တွေတလျှောက်ကနေသွားရာလမ်းကြောင်း ဖြစ်ပါတယ်။ pathname တွေကိုဖော်ပြဖို့ရာနည်းလမ်းနှစ်ခုထဲကတခုကိုအသုံးပြုနိုင်ပါတယ်။ absolute pathname တွေအနေနဲ့ သို့မဟုတ် relative pathname တွေအနေနဲ့ပေါ့။ absolute pathname တွေနဲ့အရင်လုပ်ကြည့်ကြစို့။

Absolute Pathnames

absolute pathname တခုဟာ root directory ကနေစတင်ပြီးတော့ သွားလိုတဲ့ directory သို့မဟုတ် file ဆီကိုရောက်သည့်အထိ tree သစ်ပင်ရဲ့အကိုင်းအခက်တခုကနေတခုဆီကိုလိုက်သွားတာဖြစ်ပါတယ်။ ဥပမာ - ခင်ဗျားရဲ့ system ပေါ်မှာ system program တွေတော်တော်များများကို install လုပ်ထားတဲ့ directory တခုရှိပါတယ်။ အဲ့ဒီ directory ရဲ့ pathname က /usr/bin ဖြစ်ပါတယ်။ ဆိုလိုတာက root directory (pathname ရဲ့ ရှေ့ဆုံးမှာပြထားတဲ့ slash သည် root directory ကိုကိုယ်စားပြုပါတယ်) အထဲမှာ usr ဆိုတဲ့ directory ပါဝင်ပြီးတော့ အဲ့ဒီ usr ဆိုတဲ့ directory ထဲမှာ bin ဆိုတဲ့ directory ပါဝင်နေပါတယ်။

```
jok3r@lucy:~$ cd /usr/bin
jok3r@lucy:/usr/bin$ pwd
/usr/bin
jok3r@lucy:/usr/bin$ ls
'['                               mktemp
```

2to3-2.7	mktexfmt
aa-enabled	mktexlsr
aa-exec	mktexmf
aconect	mktexpk
acpi_listen	mktextfm
add-apt-repository	mkzftree
addpart	mlabel
addr2line	mmcli
..... အောက်မှာနောက်ထပ် file ပေါင်းများစွာ list ဆက်ပြထားပါတယ်	

အခုဆိုရင်ကျွန်တော်တို့ရဲ့ current working directory ကို /usr/bin ထဲကိုပြောင်းလိုက်တာကိုကျွန်တော် တို့မြင်နေရပြီးတော့ သူ့ထဲမှာ file တွေအပြည့်ရှိနေပါတယ် ။ shell prompt ဘယ်လိုပြောင်းလဲသွားသလဲဆိုတာကို သတိပြုပါ ။ အလွယ်အနေနဲ့သုံးတာကတော့ သူ့ကို လက်ရှိရောက်နေတဲ့ working directory ရဲ့နာမည်ကိုအလို အလျောက်ဖော်ပြခိုင်းတဲ့နေရာမှာအသုံးပြုလေ့ရှိပါတယ် ။

Relative Pathnames

absolute pathname တခုက root directory ကနေစတင်ပြီးတော့ (သူသွားလိုတဲ့ file သို့မဟုတ် directory) သူ့ရဲ့အဆုံးသပ်နေရာကိုဦးဆောင်သွားခဲ့ပေမယ့် elative pathname တခုကတော့ working directory ကနေပဲစတင်ပါတယ် ။ ဒီလိုလုပ်ဖို့အတွက် သူက special symbol တချို့ကိုအသုံးပြုပြီးတော့ filesystem tree ထဲက relative position တွေကို ကိုယ်စားပြုစေပါတယ် ။ အဲ့ဒီ special symbols တွေကတော့ . (dot) နဲ့ .. (dot dot) တို့ပဲဖြစ်ကြပါတယ် ။

. symbol က working directory ကိုရည်ညွှန်း(ကိုယ်စားပြု)ပြီးတော့ . . symbol ကတော့ working directory ရဲ့ (အပေါ်က) parent directory ကိုရည်ညွှန်း(ကိုယ်စားပြု)ပါတယ် ။ သူဘယ်လိုအလုပ်လုပ်သလဲဆိုတာ ဒီမှာပြထားပါတယ် ။ နောက်တကြိမ် working directory ကို /usr/bin ပြန်ပြောင်းကြည့်ကြစို့ ။

```
jok3r@lucy:~$ cd /usr/bin
jok3r@lucy:/usr/bin$ pwd
/usr/bin
```

Okay , အခုကျွန်တော်တို့က working directory ကို /usr/bin ရဲ့ parent (directory) ဖြစ်တဲ့ /usr ကို ပြောင်းချင်တယ် ဆိုပါစို့ ။

```
jok3r@lucy:/usr/bin$ cd /usr
jok3r@lucy:/usr$ pwd
/usr
```

ဒါမှမဟုတ် relative pathname တခုနဲ့

```
jok3r@lucy:/usr/bin$ cd ..
jok3r@lucy:/usr$ pwd
/usr
```

နည်းလမ်း ၂ ခုစလုံးကတူညီတဲ့ ရလဒ်ကိုပေးပေးပါတယ်။ ။ ကျွန်တော်တို့က ဘယ်တခုကို သုံးသင့်ပါသလဲ ? စာရိုက်ရတာနည်းတဲ့နည်းကိုပဲသုံးရပါမယ် !

ထို့အတူပဲ ကျွန်တော်တို့က working directory ကို /usr ကနေ /usr/bin ထဲကို မတူညီတဲ့နည်းလမ်း ၂ ခုနဲ့ ပြောင်းမှာဖြစ်ပြီးတော့ နည်းလမ်း ၂ ခုထဲက တခုက absolute pathname ကိုသုံးမှာဖြစ်ပါတယ်။ ။

```
jok3r@lucy:/usr$ cd /usr/bin
jok3r@lucy:/usr/bin$ pwd
/usr/bin
```

ဒါမှမဟုတ် relative pathname တခုနဲ့

```
jok3r@lucy:/usr$ cd ./bin
jok3r@lucy:/usr/bin$ pwd
/usr/bin
```

အခု အဲ့ဒီမှာ အရေးကြီးတဲ့တစ်ခုကိုကျွန်တော့်အနေနဲ့ဒီနေရာမှာထောက်ပြဖို့လိုလာပါပြီ။ ။ ကိစ္စရပ်တော်တော်များများမှာ ခင်ဗျားအနေနဲ့ ./ ကိုချန်ထားခဲ့လို့ရပါတယ်။ ။ ဘာလို့လဲဆိုတော့သူကရည်ညွှန်းပြီးသား(ပါပြီးသား) မို့ပါပဲ။ ။

```
jok3r@lucy:/usr$ cd bin
```

လို့ရှိက်ထည့်လိုက်ရင်လည်း အတူတူပါပဲ ။ ယျေဘုယျအားဖြင့်တော့ ခင်ဗျားအနေနဲ့ တစ်ခုတစ်ခုဆိုသွားမယ့် pathname ကိုအတိအကျမသတ်မှတ်ထားဘူးဆိုရင် working directory ကိုပဲယူပြီးလုပ်သွားမှာဖြစ်ပါတယ် ။

Some Helpful Shortcuts

Table 2-1 ထဲမှာ ကျွန်တော်တို့အနေနဲ့ current working directory ကိုအမြန်ဆုံးပြောင်းနိုင်မယ့် အသုံးဝင်တဲ့နည်းလမ်းတချို့ကိုတွေ့ရပါလိမ့်မယ် ။

Table 2-1: cd Shortcuts

Shortcut	Result
cd	Changes the working directory to your home directory.
cd -	Changes the working directory to the previous working directory.
cd ~username	Changes the working directory to the home directory of username . For example, cd ~bob changes the directory to the home directory of user bob.

IMPORTANT FACTS ABOUT FILENAMES

- filename ရဲ့အစမှာ period character (. အစက်ကလေး) ပါရင် Hidden (ဖွက်ထားတဲ့ file) တွေဖြစ်ပါတယ် ။ ဒါက ဘာတစ်ခုထဲကိုဆိုလိုတာလဲဆိုတော့ ခင်ဗျားအနေနဲ့ ls -a လို့မရှိမချင်း ls ကနေသူတို့ကို list မလုပ်ပေး (မဖော်ပြပေး) ပါဘူး ။ ခင်ဗျားရဲ့ account ကို create လုပ်တဲ့အချိန်တုန်းက ခင်ဗျားရဲ့ account ကို configure လုပ်ဖို့အတွက် ခင်ဗျားရဲ့ home directory အထဲမှာ hidden file တွေအများအပြားကိုထားရှိခဲ့ပါတယ် ။
- Linux ထဲမှာ filename တွေနဲ့ command တွေဟာ Unix မှာလိုပဲ case sensitive (စာလုံးအကြီးအသေး ခွဲခြားဆက်ဆံခြင်း) ဖြစ်ပါတယ် ။ File1 နဲ့ file1 ဆိုတဲ့ filename နာမည် ၂ ခုကိုမတူညီတဲ့ file တွေအဖြစ်သတ်မှတ်ပါတယ် ။
- Linux ထဲမှာ အခြား operating system တွေမှာလို “file extension” နဲ့ပတ်သက်ပြီးတော့ အယူအဆမရှိပါဘူး ။ ခင်ဗျားအနေနဲ့ file တွေကိုခင်ဗျားကြိုက်သလိုနာမည်ပေးလို့ရပါတယ် ။ file တစ်ခုရဲ့ content တွေ နဲ့ ရည်ရွယ်ချက်တွေကိုအခြားအရာတွေနဲ့ပဲဆုံးဖြတ်ပါတယ် ။ Unix-like operating system တွေမှာ file

တခုရဲ့ content တွေနဲ့ ရည်ရွယ်ချက်တွေအတွက် file extension တွေကိုအသုံးပြုဘူးဆိုသည့် တိုင်အောင် တချို့ application program တွေကတော့အသုံးပြုကြပါတယ်။

- Linux မှာ embedded space တွေနဲ့ punctuation character (: ; ? ! စသည်) တွေပါဝင်တဲ့နာမည် အရှည်ကြီးတွေကို လက်ခံပေးတယ်ဆိုပေမယ့်လည်း ခင်ဗျား create လုပ်ထားတဲ့ file ရဲ့နာမည်ထဲမှာ punctuation character ဖြစ်တဲ့ period, dash (hyphen) နဲ့ underscore တွေကို limit လုပ်ထားပါတယ်။ ပိုပြီးအရေးကြီးတာကတော့ filename တွေထဲမှာ embedded space တွေမထည့်ပါနဲ့။ ကျွန်တော် တို့ Chapter 7 မှာတွေ့ရှိရမယ့်အတိုင်း filename တွေထဲမှာ embedded space တွေထည့်တာဟာ command line နဲ့လုပ်ရတဲ့အလုပ်တော်တော်များများကိုပိုပြီးခက်ခဲစေပါတယ်။ ခင်ဗျားအနေနဲ့ filename ထဲက စာလုံး ၂ လုံးကြားမှာ embedded space ထည့်ချင်တယ်ဆိုရင် underscore character (_) တွေထည့်သုံးပါ။ နောက်မှခင်ဗျားကိုခင်ဗျားကျေးဇူးတင်နေပါလိမ့်မယ်။

3

EXPLORING THE SYSTEM

အခုဆိုရင်ကျွန်တော်တို့က filesystem ထဲမှာဘယ်လိုရွေ့လျားသွားလာရတယ်ဆိုတာကိုသိသွားပြီဖြစ်တဲ့ အတွက် Linux system ကို guided tour ထွက်ဖို့အချိန်ကျလာပါပြီ။ ကျွန်တော်တို့မစခင်မှာ ၊ ဘယ်လိုပဲဖြစ်ဖြစ် ကျွန်တော်တို့အနေနဲ့လမ်းတလျှောက်မှာအသုံးဝင်မယ့် command တချို့ကိုပါလေ့လာသွားမှာဖြစ်ပါတယ်။

- ls —List directory contents. (directory ထဲမှာရှိတဲ့ content တွေကို list ထုတ်ကြည့်ခြင်း)
- file —Determine file type. (ဘာ file အမျိုးအစားလဲဆိုတာကိုကြည့်ခြင်း)
- less —View file contents. (file ထဲက စာတွေ content တွေကိုဖတ်ခြင်း)

More Fun with ls

ls ဟာအသုံးအများဆုံး command တခုဖြစ်နိုင်သလိုအဲ့ဒါအတွက်လည်းအကြောင်းပြချက်ကောင်းကောင်း ရှိပါတယ်။ အဲ့ဒါနဲ့ဆိုရင်ကျွန်တော်တို့က directory ထဲက content တွေ နဲ့ အရေးကြီးတဲ့ file မျိုးစုံနဲ့သူတို့ရဲ့ attribute တွေကိုကြည့်နိုင်ပါတယ်။ ကျွန်တော်တို့မြင်ခဲ့တဲ့အတိုင်းပဲ ကျွန်တော်တို့အနေနဲ့ ls လို့ရိုးရိုးလေးရိုက်ထည့် လိုက်ရုံနဲ့ current working directory ထဲမှာရှိတဲ့ file တွေနဲ့ subdirectorie တွေရဲ့ list ကိုမြင်နိုင်ပါတယ်။

```
jok3r@lucy:~$ ls
default.ovpn  dwhelper          mm-kb-master  Recordings
Desktop       hacking_tutorial_link  Music         snap
Documents     javasharedresources  Pictures       Templates
Downloads     master.zip          pt            Videos
Dropbox       MEGAsync           Public
jok3r@lucy:~$
```


current working directory အပြင် အတိအကျရည်ညွှန်းထားတဲ့ (ကိုယ်လှမ်းကြည့်ချင်တဲ့) directory ကို list လှမ်းခေါ်ပြီးကြည့်လို့ရပါတယ်။ ကျန်တဲ့ directory တွေလည်းအတူတူပါပဲ။

```
jok3r@lucy:~$ ls /usr
in      include lib32 libexec local share
games lib    lib64 libx32 sbin src
jok3r@lucy:~$
```

ဒါမှမဟုတ် အတိအကျရည်ညွှန်းထားတဲ့ (ကိုယ်လှမ်းကြည့်ချင်တဲ့) directory ကိုတစ်ခုထက်ပိုပြီးကြည့်နိုင်ပါတယ်။ အခု ဥပမာ ထဲမှာ ကျွန်တော်တို့အနေနဲ့ user ရဲ့ home directory (~ character နဲ့ home directory ကိုအမှတ်အသားပြုထားပါတယ်) နဲ့ /usr directory ၊ ခုစလုံးကိုတပြိုင်နက် list ထုတ်ပြမှာဖြစ်ပါတယ်။

```
jok3r@lucy:~$ ls ~ /usr
/home/jok3r:
default.ovpn  dwhelper          mm-kb-master  Recordings
Desktop       hacking_tutorial_link  Music         snap
Documents     javasharedresources  Pictures       Templates
Downloads     master.zip          pt             Videos
Dropbox       MEGAsync           Public
```

```
/usr:
bin  include lib32 libexec local share
games lib    lib64 libx32 sbin src
jok3r@lucy:~$
```

ကျွန်တော်တို့အနေနဲ့ ls ရဲ့ output ကို format ပြောင်းပြီးတော့ ပိုပြီး detail ကျတဲ့ပုံစံနဲ့ထုတ်ကြည့်လို့လည်းရပါသေးတယ်။

```
jok3r@lucy:~$ ls -l
total 1592
-rw-rw-r-- 1 jok3r jok3r    0 Jun  7 2021 default.ovpn
```

```

drwxr-xr-x 6 jok3r jok3r 4096 Apr 5 23:25 Desktop
drwxr-xr-x 9 jok3r jok3r 4096 Mar 30 01:36 Documents
drwxr-xr-x 8 jok3r jok3r 4096 Apr 5 22:32 Downloads
drwx----- 7 jok3r jok3r 4096 Jul 28 2021 Dropbox
drwxrwxr-x 2 jok3r jok3r 4096 Mar 10 21:23 dwhelper
-rw-rw-r-- 1 jok3r jok3r 786 Mar 30 02:25 hacking_tutorial_link
drwxrwxr-x 2 jok3r jok3r 4096 Jun 7 2021 javasharedresources
-rw-rw-r-- 1 jok3r jok3r 1552588 Mar 16 16:02 master.zip
drwxrwxr-x 2 jok3r jok3r 4096 Apr 3 23:38 MEGAsync
drwxrwxr-x 3 jok3r jok3r 4096 Jul 3 2021 mm-kb-master
drwxr-xr-x 2 jok3r jok3r 4096 Jun 6 2021 Music
drwxr-xr-x 2 jok3r jok3r 4096 Apr 8 00:32 Pictures
drwxrwxr-x 6 jok3r jok3r 4096 Jun 12 2021 pt
drwxr-xr-x 2 jok3r jok3r 4096 Jun 6 2021 Public
drwxr-xr-x 2 jok3r jok3r 4096 Feb 20 15:11 Recordings
drwx----- 14 jok3r jok3r 4096 Mar 13 01:07 snap
drwxr-xr-x 2 jok3r jok3r 4096 Jun 6 2021 Templates
drwxr-xr-x 3 jok3r jok3r 4096 Mar 15 21:07 Videos
jok3r@lucy:~$

```

command မှာ -l ထည့်လိုက်တဲ့အခါ ကျွန်တော်တို့အနေနဲ့ output ကို long format ပြောင်းလိုက်တာဖြစ်ပါတယ်။

Options and Arguments

ဒါက ကျွန်တော်တို့ကို command တွေအတော်များများဟာ ဘယ်လိုအလုပ်လုပ်နေတာလဲဆိုတာနဲ့ပက်သက်ပြီးတော့အရေးကြီးတဲ့ point ကိုရောက်လာစေပါတယ် ။ command တွေမှာတခါတလေသူတို့အနောက်ကနေသူတို့ရဲ့ဆောင်ရွက်ပုံတွေကိုပြောင်းလဲပေးတဲ့ option တခု သို့မဟုတ် တခုထက်ပိုပြီးပါလာတတ်ပြီးတော့ (သူ့နောက်ကနေ) နောက်ထပ် command တွေကသူ့အပေါ်မူတည်ပြီးအလုပ်လုပ်ပေးတဲ့ argument တွေတခု သို့မဟုတ် တခုထက်ပိုပြီးပါလာတတ်ပါတယ် ။ ဒါကြောင့် command တွေတော်တော်များများက အောက်ကလိုပုံစံနဲ့တူနေတတ်ပါတယ် ။

command -options arguments

command တွေတော်တော်များများက option ကိုသုံးတဲ့အခါမှာ character တလုံးထဲရဲ့အရှေ့မှာ dash တခုနဲ့ -l လိုပုံစံမျိုးနဲ့အသုံးပြုကြပါတယ် ။ GNU project ထဲကဟာတွေအပါအဝင် command တွေတော်တော်များများမှာ

စကားလုံး တခုရဲ့ရှေ့မှာ dash ၊ ခုပါတဲ့ long option ကိုလည်းပဲအထောက်အပံ့ပေး (သုံးလို့ရ) ပါတယ် ။ command တွေတော်တော်များများမှာ short option ကိုတခုထက်ပိုပြီးတော့လည်းတပြိုင်နက်လုပ်ဆောင်ပေးပါတယ် ။ အခု ဥပမာထဲမှာ ls command မှာ option ၊ ခုပေးထားပါတယ် ။ l option က long format output ကို ထုတ်ပေးမှာဖြစ်ပြီးတော့ t option ကတော့ file တွေရဲ့ modification time အပေါ်မူတည်ပြီးတော့ sort စီပေးမှာ ဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ ls -lt
```

ကျွန်တော်တို့က long option --reverse ကိုထပ်ထည့်လိုက်ပြီးတော့ sort စီထားတဲ့ order ကို ပြောင်းပြန်ဖြစ်အောင်လုပ်မှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ ls -lt --reverse
```

ls command မှာ ဖြစ်နိုင်ခြေရှိတဲ့ (သူနဲ့တွဲအလုပ်လုပ်နိုင်တဲ့) option တွေအများကြီးရှိပါတယ် ။ Table 3-1 မှာ command တော်တော်များများကို list လုပ်ပေးထားပါတယ် ။

Table 3-1: Common ls Options

Option	Long Option	Description
-a	--all	File တွေအကုန်လုံးကို list ထုတ်ပြတာပါ ။ ပုံမှန်အားဖြင့် list ထုတ်ပြလေ့မရှိတဲ့ နာမည်ရဲ့အရှေ့ဆုံးမှာ period (.) လေးတခုပါတဲ့ file တွေကိုတောင်မှထုတ်ပြပေးပါတယ် ။ (i.e., hidden).
-d	--directory	သာမန်အားဖြင့်တော့ directory တခုကို(နာမည်နဲ့)အတိအကျ ရည်ညွှန်းလိုက်တဲ့အခါမှာ ls command က directory ကိုယ်တိုင်ကို မဟုတ်ပဲ directory ထဲကပါဝင်တဲ့ content တွေကို list ထုတ်ပြမှာ ဖြစ်ပါတယ် ။ ဒီ option ကို -l option နဲ့ပေါင်းစပ်အသုံးပြုပြီးတော့ content တွေထက် directory ရဲ့ အသေးစိတ်အချက်အလက်တွေကို ထုတ်ကြည့်ရာမှာအသုံးပြုနိုင်ပါတယ် ။
-F	--classify	ဒီ option က listed လုပ်ထားတဲ့ file name တခုစီရဲ့နောက်ဆုံးမှာ indicator character တခုစီထည့်ပေးပါတယ် ။ (ဥပမာ - တကယ်လို့ directory တခုရဲ့နာမည်ဖြစ်နေခဲ့မယ်ဆိုရင် / တခုထည့်ပေးမှာပါ ။)
-h	--human-readable	long format listing ထဲမှာဆိုရင် file size တွေကို byte နဲ့မဟုတ်ပဲ human-readable format နဲ့ပြပေးမှာဖြစ်ပါတယ် ။
-l		results ကို long format နဲ့ပြပေးမှာဖြစ်ပါတယ် ။

-r	--reverse	result တွေကို reverse order သာမန်ရဲ့ပြောင်းပြန်အနေနဲ့ ပြပေးမှာ ဖြစ်ပါတယ် ။ ပုံမှန်အားဖြင့်တော့ ls ကသုံးရဲ့ result ကို အကွာစာစဉ် အရင်းစဉ်ကြီးလိုက် ascending alphabetical order နဲ့ပြပေးပါတယ် ။
-S		result တွေကို file size အပေါ်မူတည်ပြီးစီပေး ပြပေးပါတယ် ။
-t		modification time အပေါ်မူတည်ပြီးစီပေး ပြပေးပါတယ် ။

A Longer Look at Long Format

ကျွန်တော်တို့တွေ့ခဲ့ကြတဲ့အတိုင်းပဲ -l option ဟာ ls (command) ကို long format နဲ့ ဖော်ပြစေပါတယ် ။ အဲ့ဒီ format ထဲမှာ အသုံးဝင်တဲ့ information တွေအများအပြားပါဝင်ပါတယ် ။ အခုဒီမှာ Ubuntu system ထဲက directory ရဲ့ Example တွေဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ ls -l Documents/
```

```
total 956
```

```
drwxrwxr-x    2 jok3r jok3r 12288 Mar 30 01:37  Arduino_Ebooks
drwxrwxrwx    4 jok3r jok3r  4096 Mar 18 17:34  Blender_Files
-rw-rw-r--    1 jok3r jok3r 25144 Jul  7 2021  Building_Chords_Blank_Sheet.pdf
-rw-rw-r--    1 jok3r jok3r 26705 Jul  7 2021  Building_Chords.pdf
drwxrwxr-x    2 jok3r jok3r  4096 Jul 26 2021  ebook
-rw-rw-r--    1 jok3r jok3r 17951 Nov 30 17:23  English_Grammar_Basic.odt
-rw-rw-r--    1 jok3r jok3r 29981 Jul  6 2021  Interval_types.pdf
drwxrwxr-x    4 jok3r jok3r  4096 Feb 17 20:49  'Linux ebooks'
drwxrwxr-x    9 jok3r jok3r  4096 Jun  7 2021  MuseScore3
-rw-rw-r--    1 jok3r jok3r 557545 Jul 27 2021  Music_Theory_MM.odt
drwxrwxr-x    2 jok3r jok3r  4096 Mar  4 21:37  Python_Ebooks
-rw-rw-r--    1 jok3r jok3r 101416 Jun 22 2021  'Tab_intro_Firehouse_Here_For_You.jpg'
drwxrwxr-x    4 jok3r jok3r  4096 Jun 16 2021  Zoom
```

အဲ့ဒီ file တွေထဲကတစ်ခုရဲ့မတူညီတဲ့အခြား field တွေကိုကြည့်ပြီးတော့ Table 3-2 မှာသူတို့ရဲ့အဓိပ္ပါယ် (ဘာကိုဖော်ပြနေတာလဲဆိုတာ) ကိုအသေးစိတ်လေ့လာကြည့်ကြစို့ ။

Table 3-2: ls Long Listing Fields

Field	Meaning
-rw-r--r--	File ကို access လုပ်ဖို့အတွက် right တွေကိုပြထားတာဖြစ်ပါတယ် ။ ပထမဦးဆုံး character က file အမျိုးအစားကိုပြတာဖြစ်ပါတယ် ။ မတူညီတဲ့အမျိုးအစားတွေကြားမှာ d ဆိုတာက directory တခုဖြစ်ကြောင်းပြနေချိန်မှာ ထိပ်ဆုံးက dash (-) လေးက regular file ရိုးရိုးပုံမှန် file လေးတခုဖြစ်ကြောင်းပြနေတာဖြစ်ပါတယ် ။ နောက်ထပ် character ၃ ခု (rw-)က file ရဲ့ owner အတွက် Access right တွေကိုပြနေတာဖြစ်ပြီးတော့ နောက်ထပ် character ၃ ခု (r--)က file ရဲ့ group member တွေအတွက် (Access right တွေ)ပြနေတာဖြစ်ပြီးတော့ နောက်ဆုံး ၃ ခုအတွဲ (r--) က (file owner နဲ့ group member ကလွဲရင်)ကျန်တဲ့လူတွေအားလုံးအတွက် ပဲဖြစ်ပါတယ် ။
1	File ရဲ့ hard link အရေအတွက်ကိုပြတာဖြစ်ပါတယ် ။ အခု chapter ရဲ့ အဆုံးမှာ link တွေအကြောင်းဆွေးနွေးချက်ကိုကြည့်နိုင်ပါတယ် ။
jok3r	File owner ရဲ့ use name ဖြစ်ပါတယ် ။
jok3r	File ကိုပိုင်ဆိုင်တဲ့ group ရဲ့ နာမည်ဖြစ်ပါတယ် ။
4096	File ရဲ့ size ကို byte နဲ့တွက်ချက်ပြထားတာဖြစ်ပါတယ် ။
Feb 17 20:49	File ကိုနောက်ဆုံးအကြိမ် modification လုပ်ခဲ့တဲ့ date နဲ့ time ကိုပြထားတာဖြစ်ပါတယ် ။
English_Grammar_Basic.odt	File ရဲ့ နာမည်ဖြစ်ပါတယ် ။

Determining a File's Type with file

ကျွန်တော်တို့အနေနဲ့ system ကို စူးစမ်းရှာဖွေမှုတွေပြုလုပ်နေစဉ်မှာ (သူ့ထဲမှာ)ဘာ file တွေပါဝင်နေသလဲဆိုတာကိုသိရမယ်ဆိုရင်အသုံးဝင်မှာဖြစ်ပါတယ် ။ ဒီလိုပြုလုပ်ဖို့အတွက်ကျွန်တော်တို့အနေနဲ့ file ဆိုတဲ့ command ကိုအသုံးပြုပြီးတော့ file အမျိုးအစားတွေကို ဆုံးဖြတ်မှာဖြစ်ပါတယ် ။ ဥပမာ - picture.jpg လို filename တခုကို သာမန်အားဖြင့် JPEG compressed image တခုပါဝင်မယ်လို့မျှော်လင့်လို့ရပေမယ့် Linux မှာတော့ အဲ့တာမျိုးမလိုပါဘူး ။ ကျွန်တော်တို့အနေနဲ့ file command ကို ဒီလို invoke (run) လုပ်လို့ရပါတယ် ။

file filename

invoke (run) လုပ်လိုက်တဲ့အခါမှာ file command က file ထဲမှာပါဝင်နေတဲ့ content တွေရဲ့အကြောင်းကိုအကြမ်းဖျင်း (screen ပေါ်မှာ) print ထုတ်ပြမှာဖြစ်ပါတယ်။ ။ ဥပမာ -

```
jok3r@lucy:~/Pictures$ file red-fox-hero.jpg
red-fox-hero.jpg: JPEG image data, JFIF standard 1.01, aspect ratio, density 1x1, segment length 16, progressive, precision 8, 1000x660, components 3
jok3r@lucy:~/Pictures$
```

file အမျိုးအစားတွေအများကြီးရှိပါတယ်။ ။ တကယ်တော့ Linux လိုမျိုး Unix-like operating system တွေမှာ အသုံးများတဲ့ idea တွေထဲကတစ်ခုကတော့ “everything is a file.” (အရာရာကို file အဖြစ်သတ်မှတ်ပြီး လုပ်ဆောင်ပေးတဲ့) အချက်ပဲဖြစ်ပါတယ်။ ။ ကျွန်တော်တို့တွေ lesson တွေလုပ်တာများလာတာနဲ့အမျှ ဒီအချက်ဟာ ဘယ်လောက်မှန်ကန်သလဲဆိုတာကိုတွေ့လာမှာဖြစ်ပါတယ်။ ။

ခင်ဗျားရဲ့ system ပေါ်က file တွေအများစုက ဥပမာ - Mp3 နဲ့ JPEG file တွေလိုမျိုးရင်းနှီးပြီးသားဖြစ်နေစဉ်မှာ (file)အမျိုးအစားအများစုက သိသာထင်ရှားမှုအနည်းငယ်နည်းနေတတ်ပြီးတော့အများစုကတော့ တော်တော်လေးထူးဆန်းနေမှာဖြစ်ပါတယ်။ ။

Viewing File Contents with less

less command ဆိုတာ text file တွေကိုဖတ်ဖို့ program တခုပဲဖြစ်ပါတယ်။ ။ ကျွန်တော်တို့ရဲ့ Linux system တခုလုံးမှာ human-readable text ပါဝင်တဲ့ file အမြောက်အမြား ပါဝင်ပါတယ်။ ။ less program က အဲ့ဒီ file တွေကို စိစစ်ကြည့်ရှုဖို့အတွက်အဆင်ပြေတဲ့နည်းလမ်းကိုဖြည့်ဆည်းပေးထားပါတယ်။ ။

ကျွန်တော်တို့ကဘာဖြစ်လို့ အဲ့ဒီ text file တွေကိုစိစစ်ကြည့်ရှုဖို့လိုတာလဲ ? ဘာဖြစ်လို့လဲဆိုတော့ (configuration file လို့ခေါ်တဲ့) system setting တွေပါဝင်နေတဲ့ file တွေအတော်များများဟာ အဲ့ဒီ format နဲ့ သိမ်းဆည်းထားကြလို့ပါပဲ။ ။ သူတို့ကိုဖတ်ရှုနိုင်မယ်ဆိုရင် system ဘယ်လိုအလုပ်လုပ်နေတယ်ဆိုတာကို အတွင်းကျကျသိရှိနိုင်မှာဖြစ်ပါတယ်။ ။ ထပ်တိုးအနေနဲ့ကတော့ system ကနေတကယ်အသုံးပြုနေတဲ့ (script လို့ခေါ်တဲ့) program အတော်များများက အဲ့ဒီ format နဲ့ သိမ်းဆည်းထားကြပါတယ်။ ။ နောက်ပိုင်း chapter တွေမှာ ကျွန်တော်တို့က system setting တွေကို modify ပြင်ဆင်ဖြည့်စွက်တာတွေလုပ်ဖို့အတွက် text file တွေကို ဘယ်လို edit လုပ်ရမယ်ဆိုတာနဲ့ ကျွန်တော်တို့ကိုယ်ပိုင် script တွေကိုဘယ်လိုရေးသားရမယ်ဆိုတာကို လေ့လာသွားမှာဖြစ်ပါတယ်။ ။ ဒါပေမယ့်အခုတော့ ကျွန်တော်တို့က သူတို့ရဲ့ (text file တွေရဲ့) content တွေကိုပဲကြည့်ကြပါဦးမယ်။ ။

WHAT IS “TEXT”?

Computer တလုံးပေါ်မှာ information တွေကိုဖော်ပြဖို့အတွက်နည်းလမ်းပေါင်းများစွာရှိနေပါတယ် ။ နည်းလမ်းတွေအကုန်လုံးမှာ information နဲ့ သူ့ကို (information ကို) ကိုယ်စားပြုပေးမယ့်ဂဏန်းတချို့ရဲ့ ဆက်နွှယ်မှုကိုသတ်မှတ်ဆုံးဖြတ်တာတွေပါဝင်ပါတယ် ။ Computer ဟာ တကယ်တော့ ဂဏန်းတွေကိုသာနားလည်ပြီးတော့ data တွေအကုန်လုံးဟာ ကိုယ်စားပြုဂဏန်းတွေအဖြစ်အပြောင်းခံရပါတယ် ။ (Computer က data ကိုမသိပဲ number တွေကိုသာသိတဲ့အတွက်သူသိတဲ့ number တွေအဖြစ် data တွေကို convert လုပ်ပေးရတာကိုဆိုလိုတာပါ ။)

အခြားအရာတွေကရိုးရှင်းပေမယ့် တချို့သော ကိုယ်စားပြုစနစ် representation system တွေဟာအလွန်ရှုပ်ထွေးလှပါတယ် ။ (compress လုပ်ထားတဲ့ video file လိုဟာမျိုးတွေ) အစောဆုံးနဲ့အရိုးရှင်းဆုံးအရာတွေထဲကတခုကတော့ ASCII text ပဲဖြစ်ပါတယ် ။ ASCII ဆိုတာ (“As-Key” လို့အသံထွက်ပါတယ် ။) American Standard Code for Information Interchange ရဲ့ အတိုကောက်ဖြစ်ပါတယ် ။ ဒီ ရိုးရှင်းတဲ့ encoding scheme ကို Teletype machine တွေပေါ်မှာပထမဆုံးသုံးစွဲခဲ့ကြတာဖြစ်ပါတယ် ။

Text ဆိုတာ character ကနေ number ကိုရိုးရှင်းတဲ့ one-to-one mapping လုပ်ထားတာတခုဖြစ်ပါတယ် ။ အဲ့ဒါကအရမ်းသေးငယ်ကျစ်လစ်ပါတယ် ။ 50 character တွေကို 50 byte အဖြစ်ဘာသာပြန်ပေးပါတယ် ။ အဲ့ဒါက Microsoft word သို့မဟုတ် OpenOffice.org Writer လိုမျိုး Word processor document တွေမှာတွေ့ရတဲ့ text လိုဟာမျိုးတွေနဲ့မတူပါဘူး ။ အဲ့ဒီ file တွေ ၊ ရိုးရှင်းအောင်ပြောရရင် ASCII text တွေပေါ့ ၊ သူတို့ရဲ့ structure နဲ့ formatting တွေကိုဖော်ပြဖို့အတွက်သုံးတဲ့ non-text element တွေအများအပြားပါဝင်ပါတယ် ။ ရိုးရိုး Plain ASCII text file တွေမှာတော့ character တွေကိုယ်တိုင်နဲ့ tab တွေ carriage return တွေနဲ့ linefeed တွေလိုမျိုးသူတို့ကို control လုပ်တဲ့ ဟာတွေသာပါဝင်ပါတယ် ။

Linux system တခုရဲ့ အနှံ့အပြားမှာ file အများအပြားကို text format နဲ့ပဲ သိမ်းဆည်းထားလေ့ရှိပြီးတော့ Linux tool အများစုကလည်း text file တွေနဲ့အလုပ်လုပ်ပေးကြပါတယ် ။ Windows ကတောင် ဒီ format ရဲ့အရေးပါပုံကိုသိမြင်လာပါတယ် ။ (Windows မှာပါတဲ့) လူသိများတဲ့ Notepad program ဆိုတာ plain ASCII text တွေအတွက် editor တခုပဲဖြစ်ပါတယ် ။

less command ကိုဒီလိုအသုံးပြုရပါတယ် ။

less filename

စတင်လိုက်ပြီဆိုတာနဲ့ less program ကခင်ဗျားကို text file ရဲ့ အရှေ့နဲ့အနောက်ကိုအပြန်အလှန် scroll လုပ်ခွင့်ပေးပါတယ် ။ ဥပမာ - system ရဲ့ user account တွေအကုန်လုံးကိုသတ်မှတ်ထားတဲ့ file ကို edit လုပ်ဖို့ဆိုရင်အောက်ပါ command ကိုရိုက်ထည့်လိုက်ရမှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ less /etc/passwd
```


less program စတင်ပြီးဆိုတာနဲ့ကျွန်တော်တို့ file ထဲက content တွေကိုကြည့်လို့ရပါပြီ ။ တကယ်လို့ file ကစာမျက်နှာတခုထက်ပိုရှည်လျားနေမယ်ဆိုရင်ကျွန်တော်တို့အနေနဲ့ scroll up နဲ့ scroll down လုပ်ပြီးကြည့်လို့ရပါတယ် ။ less ထဲကနေထွက်ဖို့အတွက်ဆိုရင်ကျွန်တော်တို့အနေနဲ့ Q key ကိုနှိပ်လို့ရပါတယ် ။

Table 3-3 မှာ less ကနေအသုံးများတဲ့ keyboard command တချို့ကို list လုပ်ပြပေးထားပါတယ် ။

Table 3-3: less Commands

Command	Action
PAGE UP or b	စာတမျက်နှာစာအနောက်ပြန် scroll လုပ်ပေးမယ် ။
PAGE DOWN or Spacebar	စာတမျက်နှာစာအရှေ့ကို scroll လုပ်ပေးမယ် ။
Up Arrow	စာတကြောင်းစာအပေါ်တက်ပေးမယ် ။
Down Arrow	စာတကြောင်းစာအောက်ဆင်းပေးမယ် ။
G	text file ရဲ့အဆုံးကိုသွားပေးမယ် ။
1G or g	text file ရဲ့အစဦးဆုံးကိုသွားပေးမယ် ။
/characters	/နောက်မှာထည့်လိုက်တဲ့ character ပါတဲ့စာလုံးတွေအကုန် Highlight လုပ်ပြီးရှာပေးမယ် ။
n	စာတကြောင်းစာအောက်ဆင်းပေးမယ် ။
h	help screen ကိုဖော်ပြပေးပါမယ် ။ help ထဲကပြန်ထွက်ချင်ရင် q နှိပ်ပါ ။
q	Less ထဲကပြန်ထွက်ပေးပါမယ် ။

LESS IS MORE

less program ကို more လို့ခေါ်တဲ့အစောပိုင်း Unix program ကိုပိုကောင်းအောင်လုပ်ပြီးအစားထိုးဖို့ design ထုတ်ထားတာဖြစ်ပါတယ် ။ သူ့ရဲ့နာမည်ကိုစာပိုဒ်ရဲ့အပေါ်နားမှာ “less is more” ဆိုပြီးပြထားပါတယ် ။ ခေတ်သစ် architect တွေနဲ့ designer တွေရဲ့ motto (ဆိုရိုးစကား) လည်းဖြစ်ပါတယ် ။

less ဟာ စာအရှည်ကြီးတွေပါတဲ့ document တွေကို တမျက်နှာချင်းစီကြည့်လို့ရတဲ့ပုံစံနဲ့သွားတဲ့ pagers လို့ခေါ်တဲ့ program ရဲ့ class ထဲကိုကျရောက်သွားပါတယ် ။ more program အနေနဲ့စာတွေကိုအရှေ့ကိုပဲ ဆက်တိုက်ကြည့်သွားလို့ရ (နောက်ပြန်ကြည့်လို့မရ) တဲ့အချိန်မှာ less program မှာတော့စာတွေကိုအရှေ့အနောက် အပြန် အလှန် ကြည့်လို့ရတဲ့အပြင် အခြား feature တွေအများကြီးလည်းပါဝင်နေပါတယ် ။

(less နဲ့ more command အပိုင်းကိုပိုပြီးစုံစုံလင်လင်သိရအောင် ဒီ youtube video လေးကိုလည်းကြည့်ပြီး လေ့လာဖို့အကြံပြုလိုပါတယ် ။

Unix/Linux for Testers | head, tail, more & less Commands

<https://www.youtube.com/watch?v=lanbYECFFOM>)

A Guided Tour

ခင်ဗျားရဲ့ Linux system ပေါ်က filesystem layout ကအခြား Unix-like system တွေပေါ်မှာတွေ့ရတာ တွေ့နေ့အတော်လေးတူပါတယ် ။ design က Linux Filesystem Hierarchy Standard လို့ခေါ်တဲ့ published standard တခုနဲ့အတိအကျသတ်မှတ်ထားပါတယ် ။ Linux distribution တွေအကုန်လုံးက Standard ကိုမ လိုက်နာကြပေမယ့်အတော်များများကတော့ (Standard နဲ့) နီးစပ်ကြပါတယ် ။

နောက်တခုက ကျွန်တော်တို့ကိုယ်တိုင် Linux system ထဲမှာလျှောက်သွားပြီးတော့ ဘယ်ဟာကကျွန်တော် တို့ရဲ့ Linux system ကြီးကိုဖြစ်ပေါ်စေတာလဲဆိုတာကိုကြည့်ကြမှာဖြစ်ပါတယ် ။ ဒါကခင်ဗျားကို ခင်ဗျားရဲ့ navigation skill တွေလေ့ကျင့်ဖို့ရာအခွင့်အရေးတခုရစေမှာလည်းဖြစ်ပါတယ် ။ ကျွန်တော်တို့အနေနဲ့ရှာဖွေတွေ့ရှိမ ယ့်အရာတွေထဲကတချက်ကတော့ စိတ်ဝင်စားစရာကောင်းတဲ့ file တွေတော်တော်များများဟာ plain, human-readable text တွေနဲ့ဖြစ်နေတယ်ဆိုတဲ့အချက်ပါပဲ ။ ကျွန်တော်တို့ရဲ့ tour ကိုသွားရင်းနဲ့အောက်ပါအချက်တွေကို လည်း စမ်းကြည့်သွားမှာဖြစ်ပါတယ် ။

- 1. ပေးထားတဲ့ directory တွေထဲကို cd နဲ့ဝင်ကြည့်မယ် ။
- 2. directory ထဲက content တွေကို ls -l နဲ့ list ထုတ်ကြည့်မယ် ။
- 3. ခင်ဗျားအနေနဲ့စိတ်ဝင်စားစရာကောင်းတဲ့ file ကိုတွေ့တဲ့အခါ ဘာ file လဲဆိုတာကို file command နဲ့စစ် ကြည့်မယ် ။
- 4. တကယ်လို့အဲ့ဒါကိုကြည့်ရတာ text file နဲ့တူနေမယ်ဆိုရင် less နဲ့ဖွင့်ကြည့်မယ် ။

Note: copy-and-paste trick ကိုသတိရလိုက်ပါ ။ တကယ်လို့ခင်ဗျားက mouse နဲ့သုံးနေတာ ဆိုရင် copy လုပ်ဖို့အတွက် file name ကို double click လုပ်ပြီးတော့ command ထဲကို paste ချဖို့အတွက် middle click ကိုနှိပ်ရမှာဖြစ်ပါတယ် ။

ကျွန်တော်တို့ (file system ထဲ) လျှောက်သွားကြည့်နေစဉ်မှာအရာဝတ္ထုတွေ (file တွေ folder တွေ) ကို ကြည့် ရှုဖို့အတွက်မကြောက်ပါနဲ့ ။ အရာဝတ္ထုတွေ (file တွေ folder တွေ) ကိုregular user တွေ နောက်ယှက် ဖျက်ဆီးခြင်းမှကျယ်ကျယ်ပြန့်ပြန့်တားမြစ်ထားပါတယ် ။ ဒါကလည်း system administrator ရဲ့အလုပ်ပဲဖြစ်ပါတယ် ။ တကယ်လို့ command တခုက တစုံတခုသောအကြောင်းအတွက် complain တက်နေတယ်ဆိုရင်နောက်တခုကို

ပြောင်းလုပ်လိုက်ပါ ။ အချိန်တစ်ခုယူပြီးတော့(file system ထဲမှာ)လှည့်ပတ်ကြည့်ရှုလိုက်ပါ ။ system ဆိုတာ ကျွန်တော်တို့စူးစမ်းလေ့လာဖို့အတွက်ပဲဖြစ်ပါတယ် ။ တစ်ခုမှတ်ထားရမှာက Linux မှာ လျှို့ဝှက်ချက်ဆိုတာမရှိပါဘူး ။

Table 3-4 မှာကျွန်တော်တို့စူးစမ်းလေ့လာလို့ရမယ့် directory တချို့ကို list ထုတ်ပြထားပါတယ် ။ နောက်ထပ်အရာတွေကိုလည်းလွတ်လပ်စွာထပ်မံလေ့လာခွင့်ရှိပါတယ် ။

Table 3-4: Directories Found on Linux Systems

Directory	Comments
/	root directory ဖြစ်ပါတယ် ။ အရာအားလုံးစတင်ရာလည်းဖြစ်ပါတယ် ။
/bin	System boot ပြီး run ဖို့အတွက်ရှိနေရမယ့် binary (program)တွေပါဝင်ပါတယ် ။
/boot	Linux kernel , initial RAM disk image (boot time မှာလိုအပ်တဲ့ driver တွေအတွက်) ပြီးတော့ boot loader တို့ပါဝင်ပါတယ် ။
/dev	ဒါကတော့ device node တွေပါဝင်တဲ့ special directory ဖြစ်ပါတယ် ။ “Everything is a file” အရာရာဟာ file တွေဖြစ်တယ်ဆိုတဲ့အချက်ဟာ device တွေအပေါ်ကိုလည်းသက်ရောက်ပါတယ် ။ ဒီနေရာမှာ kernel က သူနားလည်သမျှ device တွေအကုန်လုံးကို list တစ်ခုနဲ့ထိန်းသိမ်းထားတဲ့နေရာလည်းဖြစ်ပါတယ် ။
/etc	<p>/etc directory မှာ system တစ်ခုလုံးက configuration file တွေပါဝင်ပါတယ် ။ သူက boot time မှာ system service တွေတစ်ခုချင်းစီကိုစတင်ပေးတဲ့ shell script တွေကိုစုစည်းပေးထားတာတစ်ခုလည်းပါဝင်ပါသေးတယ် ။ ဒီ directory ထဲမှာရှိသမျှအရာအားလုံးဟာ ဖတ်လို့ရတဲ့ readable text တွေဖြစ်ရပါမယ် ။</p> <p>စိတ်ဝင်စားစရာကောင်းတဲ့ file များ : : /etc ထဲမှာရှိသမျှအရာအားလုံးကစိတ်ဝင်စားစရာကောင်းပေမယ့်လည်းကျွန်တော့်ရဲ့အမြဲတမ်းအနှစ်သက်ဆုံးတွေကတော့ဒီမှာပါ ။</p> <ul style="list-style-type: none"> • /etc/crontab, automated job တွေ run မယ့်အခါမှာပြဌာန်းပေးတဲ့ file တစ်ခုပါ ။ • /etc/fstab, storage device တွေနဲ့သူတို့နဲ့ပတ်သက်တဲ့ mount point တွေပါဝင်တဲ့ table တစ်ခုပါ ။ • /etc/passwd, user account တွေရဲ့ list တစ်ခုပါ ။
/home	normal configuration တွေထဲမှာဆိုရင် user တယောက်ချင်းစီအတွက် directory တစ်ခုကို /home ထဲမှာပေးထားလေ့ရှိပါတယ် ။ ရိုးရိုး Ordinary user တွေမှာ file တွေကိုသူတို့ရဲ့ home directorie ထဲမှာသာ ရေးသား write လုပ်ခွင့်ရှိပါတယ် ။ ဒီလိုကန့်သတ်ချက်တွေက မူမမှန်တဲ့ user တွေရဲ့လုပ်ဆောင်ချက်တွေကနေ system ကိုကာကွယ်ပေးထားပါတယ် ။

/lib	Core system program တွေကနေအသုံးပြုနေတဲ့ share library တွေပါဝင်ပါတယ် ။ အဲ့ဒါတွေက windows ထဲက DLL တွေနဲ့ဆင်တူပါတယ် ။
/lost+found	Linux file system ကိုသုံးတဲ့ format ရိုက်ပြီးသား partation သို့မဟုတ် device တွေ ၊ ext3 လိုမျိုးဟာတွေကဒီ directory ရှိကိုရှိရပါလိမ့်မယ် ။ အဲ့ဒါတွေက file system ပျက်သွားတဲ့အချိန်မျိုးမှာ တစ်စိတ်တစ်ပိုင်းပြန်လည် recovery ရယူတဲ့ကိစ္စမျိုးမှာအသုံးပြုပါတယ် ။ ခင်ဗျားရဲ့ system မှာ တကယ့်ကိုဆိုးဆိုးရွားရွားတခုခုမဖြစ်ခဲ့ဘူးဆိုရင်တော့ ဒီ directory ကအလွတ်ကြီးပဲဖြစ်နေမှာပါ ။
/media	ခေတ်သစ် Linux system တွေမှာ /media directory မှာ USB drive တွေ CD-ROM တွေလိုထည့်လိုက်တာနဲ့အလိုအလျောက် mount တက်ပေးပြီး ဖြုတ်/တပ်လို့ရတဲ့removable media အတွက် mount point တွေပါဝင်ပါလိမ့်မယ် ။
/mnt	Linux system အဟောင်းတွေမှာဆိုရင် /mnt directory ထဲမှာ manually mount လုပ်ပေးရတဲ့ ဖြုတ်/တပ်လို့ရတဲ့ removable media အတွက် mount point တွေပါဝင်ပါလိမ့်မယ် ။
/opt	Optional software တွေကို install လုပ်တဲ့အခါမှာ /opt directory ကိုအသုံးပြုပါတယ် ။ သူကအဓိကအားဖြင့် ခင်ဗျားရဲ့ system ပေါ်မှာ install လုပ်ချင်လုပ်မှာဖြစ်တဲ့ commercial software product တွေကို hold ကိုင်ထားပေးလေ့ရှိပါတယ် ။
/proc	/proc directory ကထူးခြားပါတယ် ။ သူက ခင်ဗျားရဲ့ hard drive ပေါ်မှာ file တွေကို store လုပ်ထားတဲ့သဘောတရားမျိုးရှိတဲ့တကယ့် file system တခုမဟုတ်ပါဘူး ။ သူက Linux kernel ကနေထိမ်းသိမ်းပေးထားတဲ့ virtual filesystem တခုသာဖြစ်ပါတယ် ။ သူ့မှာပါဝင်နေတဲ့ file တွေဟာ kernel ထဲကိုကြည့်တဲ့ ချောင်းကြည့်ပေါက် peephole တွေပါဝင်ပါတယ် ။ ဒီ file တွေ ဟာ ဖတ်လို့ရတဲ့ readable တွေဖြစ်ပြီးတော့ ခင်ဗျားရဲ့ computer ကို kernel က ဘယ်လိုရှုမြင်သလဲဆိုတာကိုမြင်ယောင်စေမှာဖြစ်ပါတယ် ။
/root	ဒါကတော့ root account ရဲ့ home directory ပဲဖြစ်ပါတယ် ။
/sbin	ဒီ directory ထဲမှာ system ရဲ့ binary တွေပါဝင်ပါတယ် ။ ဒါတွေက ယျေဘုယျအားဖြင့် superuser အတွက်သီးသန့်ထားတဲ့ vital system task တွေကိုလုပ်ဆောင်ပေးတဲ့ program တွေပဲဖြစ်ပါတယ် ။
/tmp	/tmp directory ကိုတော့ program ပေါင်းစုံကနေ ဖန်တီးလိုက်တဲ့ မတည်မြဲပျောက်ကွယ်သွားမည့် file တွေကို ယာယီသိုမှီးထားရန်အတွက်ရည်ရွယ်ထားပါတယ် ။ system reboot တက်တဲ့အချိန်တိုင်းမှာ တချို့ configuration တွေက ဒီ directory ကို အလွတ်ကြီးဖြစ်သွားစေပါတယ် ။
/usr	/usr directory tree ကတော့ Linus system တခုမှာအကြီးဆုံးဖြစ်ဖို့များပါတယ် ။ သူ့မှာ regular user ကနေအသုံးပြုနေတဲ့ program တွေနဲ့ support file တွေအကုန်လုံးပါဝင်ပါတယ် ။

/usr/bin	/usr/bin ထဲမှာ ခင်ဗျားရဲ့ Linux distribution ကနေအသုံးပြုနေတဲ့ executable program တွေကို install လုပ်ထားပါတယ် ။ ဒီ directory အတွက်ကတော့ program တွေ ထောင်နဲ့ချီပြီး ကိုင်တွယ်ထားတာက သာမန်မဟုတ်တာမျိုး မဟုတ်ပါဘူး ။
/usr/lib	/usr/bin ထဲက program တွေအတွက် shared library တွေဖြစ်ပါတယ် ။
/usr/local	/usr/local tree ဆိုတာ ခင်ဗျားရဲ့ distribution ထဲမှာမပါဝင်ပေမယ့် system တခုလုံးမှာ အသုံးပြုဖို့ရည်ရွယ်ထားတဲ့ program တွေရှိတဲ့နေရာဖြစ်ပါတယ် ။ source code ကနေ compile လုပ်လိုက်တဲ့ program တွေဟာ သာမန်အားဖြင့် /usr/local/bin ထဲမှာ install လုပ် ထားလေ့ရှိပါတယ် ။ အသစ်ချက်ချွတ် install လုပ်ထားတဲ့ Linux system တခုမှာဆိုရင် ဒီ tree ရှိပါတယ် ။ ဒါပေမယ့် system administrator ကနေတခုခုသွားမထည့်မချင်းသူက အလွတ် ကြီးပဲရှိနေမှာဖြစ်ပါတယ် ။
/usr/sbin	system administration program တွေနောက်ထပ်ပါဝင်ပါတယ် ။
/usr/share	/usr/share ထဲမှာ /usr/bin ထဲက program တွေသုံးတဲ့ shared data တွေအကုန်လုံးပါဝင်ပါ တယ် ။
/usr/share/doc	System ပေါ်မှာ install လုပ်သမျှ package တွေအများစုမှာ documentation တမျိုးမျိုးပါဝင် နေတတ်ပါတယ် ။ /usr/share/doc ထဲမှာ package (နာမည်)အလိုက်စုစည်းပေးထားတဲ့ documentation file တွေကိုကျွန်တော်တို့ရှာဖွေတွေ့ရှိနိုင်ပါတယ် ။
/var	/tmp နဲ့ /home ကလွဲရင် အခုချိန်အထိကျွန်တော်တို့ကြည့်ခဲ့သမျှ directory တွေအကုန်လုံး ဟာ static အငြိမ်တွေများပါတယ် ။ ဒါပဲလေ ။ သူတို့ရဲ့ content တွေကပြောင်းလဲမှုမရှိပါဘူး ။ /var directory tree မှာ ပြောင်းလဲနေဖို့များတဲ့ data တွေကိုသိုလှောင်ထားမှာဖြစ်ပါတယ် ။ database တွေမျိုးစုံ၊ spool file တွေ၊ user mail စတာတွေကိုဒီမှာရှာတွေ့နိုင်ပါတယ် ။
/var/log	/var/log ထဲမှာ system activity တွေမျိုးစုံပါဝင်တဲ့ log file တွေပါဝင်ပါတယ် ။ ဒါတွေက အရမ်းအရေးကြီးပြီးတော့အချိန်တိုင်းစောင့်ကြည့်နေသင့်ပါတယ် ။ အသုံးအဝင်ဆုံးတခုကတော့ /var/log/messages ပဲဖြစ်ပါတယ် ။ လုံခြုံရေးအကြောင်းပြချက်အရ တခုမှတ်ထားသင့်တာက ခင်ဗျားအနေနဲ့ log file တွေကိုကြည့်ဖို့အတွက် superuser ဖြစ်မှရပါမယ် ။

(linux file system နဲ့ပတ်သက်ပြီးပိုမိုနားလည်စေဖို့ youtube link လေးတွေထည့်ပေးလိုက်ပါတယ် ။ ကြည့်ဖြစ် အောင်ကြည့်စေချင်ပါတယ် ။

Linux Directories Explained in 100 Seconds

<https://www.youtube.com/watch?v=42iQKuQodW4>

https://www.youtube.com/watch?v=e56dgj9_sSE

the Linux File System explained in 1,233 seconds // Linux for Hackers // EP 2

<https://www.youtube.com/watch?v=A3G-3hp88mo>)

Symbolic Links

ကျွန်တော်တို့ လျှောက်ကြည့်ရင်းနဲ့ ဒီလိုပုံစံမျိုး directory listing တခုကိုတွေ့မြင်နိုင်ဖို့များပါတယ်။

```
lrwxrwxrwx 1 root root 11 2012-08-11 07:34 libc.so.6 -> libc-2.6.so
```

list ထဲကပထမဦးဆုံးစာလုံးက l ဘယ်လိုဖြစ်နေတာလဲဆိုတာနဲ့ ထည့်ထားတဲ့ filename က j ခုလိုဖြစ်နေတာကိုသတိထားကြည့်လိုက်ပါ။ ဒီဟာကထူးခြားတဲ့ file အမျိုးအစားဖြစ်ပြီးတော့သူ့ကို symbolic link (soft link သို့မဟုတ် symlink လို့လည်းသိကြပါတယ်။) လို့ခေါ်ပါတယ်။ Unix-like system တွေမှာ နာမည်တွေအများအပြားနဲ့ reference လုပ်ထားတဲ့ file တခုရှိဖို့ဖြစ်နိုင်ပါတယ်။ သူ့ရဲ့တန်ဖိုးကအခုတော့သိပ်မသိသာသေးပေမယ့်ဒီဟာကတကယ့်ကိုအသုံးဝင်တဲ့ feature တခုဖြစ်ပါတယ်။

ဒီဖြစ်စဉ်ကိုမျက်စိထဲမြင်ယောင်ကြည့်ပါ : program တခုဟာ foo လို့ခေါ်တဲ့ file ထဲမှာရှိတဲ့ share resource လိုဟာမျိုးကိုအသုံးပြုဖို့လိုအပ်နေပါတယ်။ ဒါပေမယ့် foo ဟာလက်ရှိ version မှာအပြောင်းအလဲဖြစ်သွားပါတယ်။ (version ပြောင်းသွားပါတယ်။) filename ထဲမှာ version number ထည့်လိုက်ရင်ကောင်းမှာပဲ။ ဒါမှသာ administrator သို့မဟုတ် အခြားစိတ်ဝင်စားတဲ့အဖွဲ့တွေက foo ဟာ ဘယ် version ကို install လုပ်ထားသလဲဆိုတာကိုသိကြမှာဖြစ်ပါတယ်။ ဒါက ပြဿနာဖြစ်ပေါ်စေပါတယ်။ တကယ်လို့ကျွန်တော်တို့က share resource ရဲ့နာမည်ကိုပြောင်းလိုက်မယ်ဆိုရင် resource ရဲ့နာမည်အသစ်ပြောင်းလိုက်တဲ့အချိန်တိုင်းမှာ ကျွန်တော်တို့အနေနဲ့သူ့ကို (foo ကို) အသုံးပြုနေတယ်လို့ထင်ရတဲ့ program မှန်သမျှကိုခြေရာခံလိုက်ရမှာဖြစ်ပြီးတော့ resource ရဲ့နာမည်အသစ်ကိုလိုက်ရှာဖို့ခိုင်းနေရမှာဖြစ်ပါတယ်။ ဒါကသိပ်တော့ပျော်စရာမကောင်းဘူးလေ။

အဲ့ဒီမှာ symbolic link တွေက(ခင်ဗျား)ကိုကယ်တင်လိုက်တာပါပဲ။ ဆိုကြပါစို့ ကျွန်တော်တို့က foo ရဲ့ version 2.6 ကို install လုပ်တယ်ပေါ့။ သူ့ filename က foo-2.6 ပေါ့။ ဒါကဘာကိုဆိုလိုသလဲဆိုတော့ program တခုက foo ဆိုတဲ့ file ကိုဖွင့်တဲ့အခါမှာတကယ်တော့သူက foo-2.6 ဆိုတဲ့ file ကိုသွားဖွင့်တာဖြစ်ပါတယ်။ အခုဆိုရင်အကုန်လုံးပျော်ရပြီပေါ့။ foo ပေါ်မှာမှီခိုနေရတဲ့ program ကလည်းသူ့ကိုရှာတွေ့နိုင်ပြီးတော့ ကျွန်တော်တို့အနေနဲ့လည်း ဘယ် version ကိုသွင်းထားသလဲဆိုတာကိုအတိအကျသိနိုင်သွားပါပြီ။ foo-2.7 ကို upgrade လုပ်ဖို့အချိန်ကျတဲ့အခါကျရင်လည်း ကျွန်တော်တို့ system ထဲမှာ file (အသစ်)ကိုပေါင်းထည့်လိုက်မယ်။ symbolic link ကိုဖျက်လိုက်ပြီးတော့ version အသစ်ဆီကိုညွှန်ပြနေတဲ့ symbolic link အသစ်တခုထပ်ဆောက်ပေးလိုက်ရုံပါပဲ။

ဒါက version upgrade နဲ့ပတ်သက်တဲ့ပြဿနာကိုဖြေရှင်းပေးရုံတင်မကပဲကျွန်တော်တို့စက်ပေါ်မှာ version ၂ ခု စလုံးကိုထားခွင့်ပေးလိုက်ပါတယ် ။ foo-2.7 မှာ bug ရှိနေတယ် (သောက် developer တွေက) ပြီးတော့ ကျွန်တော်တို့က အရင် version အဟောင်းကိုပြန်ပြောင်းရမယ်လို့စိတ်ကူးယဉ်ကြည့်ပါ ။ နောက်တကြိမ် ထပ်ပြီး တော့ကျွန်တော်တို့က new version ဆီကိုညွှန်ပြနေတဲ့ symbolic link ကိုထပ်ဖျက်ပြီးတော့ version အဟောင်းဆီ ကိုညွှန်ပြနေတဲ့ symbolic link အသစ်တခုကိုဖန်တီးပေးရမှာဖြစ်ပါတယ် ။

အပေါ်မှာပြထားခဲ့တဲ့ directory listing (Fedora system ရဲ့ /lib directory ကနေယူထားတာပါ) က share library file တခုဖြစ်တဲ့ libc-2.6.so ကိုညွှန်ပြနေတဲ့ libc.so.6 လို့ခေါ်တဲ့ symbolic link ကိုဖော်ပြနေတာဖြစ် ပါတယ် ။ နောက် chapter မှာကျွန်တော်တို့အနေနဲ့ symbolic link တွေကိုဘယ်လို ဖန်တီးရသလဲဆိုတာကို လေ့လာကြမှာဖြစ်ပါတယ် ။

HARD LINKS

ကျွန်တော်တို့အနေနဲ့ link တွေအကြောင်းပြောနေတုန်း ကျွန်တော်တို့မှာဒုတိယ link အမျိုးအစာတခုဖြစ် တဲ့ Hard link ရှိကြောင်းထုတ်ပြောဖို့လိုအပ်လာပါတယ် ။ Hard link ကလည်း file တွေကိုနာမည်အများကြီးရှိခွင့် ပေးပါတယ် ။ ဒါပေမယ့်သူတို့ကမတူညီတဲ့အခြားနည်းလမ်းတခုနဲ့လုပ်ပေးပါတယ် ။ ကျွန်တော်တို့အနေနဲ့ symbolic နဲ့ hard link တွေကြားကွာခြားချက်ကိုနောက် chapter ကျရင်ပြောသွားမှာဖြစ်ပါတယ် ။

4

MANIPULATING FILES AND DIRECTORIES

ဒီအတိုင်းဆိုရင်ကျွန်တော်တို့က တကယ့်အလုပ်တချို့ကိုလုပ်ဖို့အဆင်သင့်ဖြစ်နေပါပြီ ။ အခု chapter က အောက်ပါ command တွေကိုမိတ်ဆက်ပေးသွားမှာဖြစ်ပါတယ် ။

- cp — file နဲ့ directory တွေကို copy လုပ်ပေးမယ် ။
- mv —file နဲ့ directory တွေကို move သို့မဟုတ် rename လုပ်ပေးမယ် ။
- mkdir — directory ဖန်တီးပေးမယ် ။
- rm — file နဲ့ directory တွေကို remove လုပ်ပေးမယ် ။
- ln — hard link နဲ့ symbolic link တွေဖန်တီးပေးမယ် ။

ဒီ command ၅ ခုဟာ Linux command တွေထဲမှာ မကြာခဏအသုံးအပြုဆုံး command တွေဖြစ်ပါတယ် ။ သူတို့တွေကို file နဲ့ directory နှစ်မျိုးစလုံးကို manipulating လုပ်တဲ့နေရာမှာသုံးပါတယ် ။

အခု ၊ ပွင့်ပွင့်လင်းလင်းပြောရမယ်ဆိုရင်တော့ command ကနေလုပ်ပေးတဲ့ task တချို့ဟာ graphical file manager ကနေလုပ်ရင်ပိုပြီးလွယ်ကူပါတယ် ။ file manager တခုနဲ့ဆိုရင်ကျွန်တော်တို့က file တွေကို directory တခုကနေ နောက်တခုဆီကို drag and drop ၊ file တွေကို cut and paste ၊ file တွေကို delete စသည် ဖြင့်ပြုလုပ်လို့ရပါတယ် ။ ဒါဆိုဘာလို့ command-line program အဟောင်းကိုသုံးနေတာလဲ ?

အဖြေကတော့ power and flexibility ကြောင့်ပါပဲ ။ ရိုးရှင်းတဲ့ file manipulation တွေကို graphical file manager တခုနဲ့အလွယ်တကူလုပ်ကိုင်နိုင်နေချိန်မှာ ခက်ခဲတဲ့အလုပ်တွေကို command-line program တွေနဲ့ အလွယ်တကူလုပ်ကိုင်နိုင်ပါတယ် ။ ဥပမာ - ကျွန်တော်တို့အနေနဲ့ directory တခုကနေနောက်တခုဆီကို HTML file

တွေအကုန်လုံးကို copy လုပ်ချင်တယ် ၊ ဒါပေမယ့် (HTML file တွေ)သွားထားမယ့် directory ထဲမှာမရှိသေးတဲ့ file တွေ ဒါမှမဟုတ် (HTML file တွေ)သွားထားမယ့် directory ထဲက file တွေထက် version ပိုမြင့်တာတွေကိုပဲ copy လုပ်ချင်တယ် ဆိုရင် file manager တခုနဲ့သွားလုပ်ဖို့အတော်ခက်ခဲပါတယ် ။ command line နဲ့သွားလုပ်ရင်တော့ အတော်လွယ်ကူပါတယ် ။

```
cp -u *.html destination
```

Wildcards

ကျွန်တော်တို့အနေနဲ့ကျွန်တော်တို့ရဲ့ command တွေကို စတင် အသုံးပြုခင်မှာ ကျွန်တော်တို့အနေနဲ့ command တွေကိုအရမ်း powerful ဖြစ်စေတဲ့ shell feature တွေအကြောင်းကိုပြောဖို့လိုအပ်ပါတယ် ။ ဘာလို့လဲဆိုတော့ shell က file name တွေကိုအရမ်းအသုံးပြုလွန်းတာကြောင့် သူက file name တွေကိုသတ်မှတ်ထားတဲ့ group အနေနဲ့ထုတ်ပေးတဲ့နေရာမှာခင်ဗျားကိုကူညီဖို့အတွက် special character တွေထည့်ပေးထားပါတယ် ။ အဲ့ဒီ special character တွေကို wildcard တွေလို့ခေါ်ပါတယ် ။ wildcard (globbing လို့လည်းခေါ်ကြပါတယ်) တွေကိုအသုံးပြုခြင်းအားဖြင့်ခင်ဗျားကို character တွေရဲ့ pattern အပေါ်မူတည်ပြီးတော့ file name တွေကိုရွေးချယ်တာကိုခွင့်ပြု (ပြုလုပ်)ပေးပါတယ် ။ Table 4-1 မှာ wildcard တွေရဲ့ list နဲ့ သူတို့ဘာကို select လုပ်ပေးတယ်ဆိုတာကိုပြပေးထားပါတယ် ။

Table 4-1: Wildcards

Wildcard	Matches
*	ဘယ်လို character ကိုမဆို
?	ဘယ်လို single character ကိုမဆို
[characters]	set character တွေရဲ့ member ဖြစ်တဲ့ ဘယ်လို character ကိုမဆို
[!characters]	set character တွေရဲ့ member မဟုတ်တဲ့ ဘယ်လို character ကိုမဆို
[[:class:]]	special class ရဲ့ member ဖြစ်တဲ့ ဘယ်လို character ကိုမဆို

Table 4-2 မှာ အသုံးအများဆုံး character class တွေကို list ထုတ်ပြထားပါတယ် ။

Table 4-2: Commonly Used Character Classes

Character Class	Matches
-----------------	---------

[alnum:]	မည်သည့် alphanumeric character ကိုမဆို
[alpha:]	မည်သည့် alphabetic character ကိုမဆို
[digit:]	မည်သည့် numeral ကိုမဆို
[lower:]	မည်သည့် စာလုံးအသေး lower case ကိုမဆို
[upper:]	မည်သည့် စာလုံးအကြီး upper case ကိုမဆို

wildcard တွေကိုအသုံးပြုခြင်းအားဖြင့် filename တွေအတွက်အလွန်ခက်ခဲရှုပ်ထွေးတဲ့ selection ဆုံးဖြတ်ချက်များကို ဖြစ်နိုင်အောင်လုပ်ပေးလာနိုင်ပါတယ် ။ Table 4-3 မှာ pattern တွေရဲ့ ဥပမာတချို့နဲ့သူတို့က ဘာတွေကိုယှဉ်တွဲ (ရှာဖွေ) ပေးတယ်ဆိုတာကို list ထုတ်ပြထားပါတယ် ။

Table 4-3: Wildcard Examples

Pattern	Matches
*	File တွေအကုန်လုံး
g*	g နဲ့စတဲ့ ဘယ် file ကိုမဆို
b*.txt	b နဲ့စပြီးသူ့အနောက်ကဘာစာလုံးတွေပဲပါပါ ၊ ပြီးတော့ .txt နဲ့ဆုံးတဲ့ ဘယ် file ကိုမဆို
Data???	Data ဆိုတာနဲ့စပြီးတော့သူ့အနောက်က character ၃ လုံးပဲပါတဲ့ ဘယ် file ကိုမဆို
[abc]*	a, b သို့မဟုတ် c နဲ့စတဲ့ ဘယ် file ကိုမဆို
BACKUP.[0-9][0-9][0-9]	BACKUP နဲ့စတဲ့ ဘယ် file ကိုမဆို ။ သူ့အနောက်မှာ numeral ၃ လုံးပဲပါရမယ် ။
[[:upper:]]*	uppercase letter နဲ့စတဲ့ ဘယ် file ကိုမဆို
[[:digit:]]*	numeral တလုံးနဲ့ စတဲ့ ဘယ် file ကိုမဆို
*[[:lower:]]123]	lowercase letter တလုံး သို့မဟုတ် numeral 1,2 သို့မဟုတ် 3 နဲ့ဆုံးတဲ့ ဘယ် file ကိုမဆို

filename တွေကို argument တွေအနေနဲ့လက်ခံတဲ့မည်သည့် command နဲ့မဆို wildcard တွေကို အသုံးပြုလို့ရပါတယ် ။ ဒါပေမယ့် ကျွန်တော်တို့ Chapter ကျရင်ဒီအကြောင်းကိုပိုပြောပါဦးမယ် ။

CHARACTER RANGES

တကယ်လို့ခင်ဗျားက Unix-like environment တခုကနေလာတာ ဒါမှမဟုတ် ဒီအကြောင်းအရာ ဘာသာရပ်နဲ့ပတ်သက်ပြီးတော့ဖတ်ဖူးတာရှိရင်ခင်ဗျားအနေနဲ့ [A-Z] သို့မဟုတ် [a-z] character range notation တွေနဲ့ကြုံတွေ့ဖူးမှာဖြစ်ပါတယ် ။ ဒါတွေက Unix notation တွေဖြစ်ပြီးတော့သူတို့က Linux version အဟောင်းတွေ ထဲမှာလည်းအလုပ်လုပ်ကြပါတယ် ။ သူတို့က (အခုတိုင်အောင်) အလုပ်လုပ်နေနိုင်ပါတယ် ဒါပေမယ့်ခင်ဗျားအနေနဲ့

သူတို့ကို (သုံးတဲ့အခါမှာ) အလွန်သတိထားရမှာဖြစ်ပါတယ် ။ ဘာလို့လဲဆိုတော့ခင်ဗျားအနေနဲ့မှန်မှန်ကန်ကန် configure မလုပ်နိုင်ရင် သူတို့ကခင်ဗျားမျှော်လင့်ထားသလိုအဖြေထွက်လာမှာမဟုတ်လို့ဖြစ်ပါတယ် ။

WILDCARDS WORK IN THE GUI TOO

wildcard တွေဟာအထူးတလည်အဖိုးတန်ကြပါတယ် ။ ခင်ဗျားအနေနဲ့ command-line ပေါ်မှာမကြာခဏ သုံးနေတာကြောင့်တင်မကဘူး သူတို့ကို graphical file manager တချို့ကပါ support ပေးတဲ့အတွက်ကြောင့်ဖြစ်ပါတယ် ။

- Nautilus (Gnome ရဲ့ file manager) ထဲမှာဆိုရင်ခင်ဗျားအနေနဲ့ file တွေကို Edit > Select Pattern မှာ select သွားလုပ်လို့ရပါတယ် ။ file selection pattern တခုနဲ့အတူ wildcard တွေကိုထည့်ပေးလိုက်ရင် လက်ရှိကြည့်နေတဲ့ directory ထဲက file တွေက selection အတွက် highlight ဖြစ်သွားမှာဖြစ်ပါတယ် ။
- Dolphin နဲ့ Konqueror (KDE ရဲ့ file manager) တို့ရဲ့ တချို့ version တွေမှာဆိုရင် ခင်ဗျားအနေနဲ့ wildcard တွေကို location bar မှာတိုက်ရိုက်ထည့်လို့ရပါတယ် ။ ဥပမာ - တကယ်လို့ခင်ဗျားအနေနဲ့ /usr/bin directory ထဲက lowercase u နဲ့စတဲ့ file တွေအကုန်လုံးကိုကြည့်ချင်တယ်ဆိုရင် ခင်ဗျားအနေနဲ့ location bar မှာ /usr/bin/u* လို့ရိုက်ထည့်လိုက်တာနဲ့သူက result ကိုပြပေးမှာဖြစ်ပါတယ် ။

command line interface မှာနဂိုမူလတွေ့ရတဲ့စိတ်ကူးတော်တော်များများကို graphical interface ပေါ်မှာလည်းပဲတနည်းနည်းနဲ့ရောက်ရှိသွားပါတယ် ။ အဲ့ဒါက Linux desktop ကို so powerful ဖြစ်စေတဲ့အချက်တွေ အများကြီးထဲကတချက်လည်းဖြစ်ပါတယ် ။

mkdir—Create Directories

mkdir command ကို directory တွေကိုတည်ဆောက်ဖန်တီးတဲ့နေရာမှာအသုံးပြုပါတယ် ။ သူကဒီလို အလုပ်လုပ်ပါတယ် ။

mkdir directory...

A note on notation: ယခုစာအုပ်ထဲမှာ command တခုထဲက argument တခုရဲ့အနောက်မှာ period အစက် ၃ စက်ပါလာတဲ့အခါမှာ (အပေါ်မှာပြခဲ့တဲ့အတိုင်း) သူကဘာကိုဆိုလိုသလဲဆိုတော့ အဲ့ဒီ argument ဟာ ထပ်ခါ ထပ်ခါ repeat လုပ်လို့ရပါတယ်တဲ့ ။ ဒါကြောင့် အခုကိစ္စမှာ

mkdir dir1

ဆိုတဲ့ command က dir1 လို့အမည်ပေးထားတဲ့ directory တခုထဲကိုတည်ဆောက်ဖန်တီးနေစဉ်မှာ

mkdir dir1 dir2 dir3

ဆိုတဲ့ command က dir1, dir2 နဲ့ dir3 လို့အမည်ပေးထားတဲ့ directory တွေကို တည်ဆောက်ဖန်တီးပေးမှာဖြစ်ပါတယ် ။

cp—Copy Files and Directories

cp command က file သို့မဟုတ် directory တွေကို copy ကူးပေးပါတယ် ။ သူ့ကိုနည်းလမ်း ၂ မျိုးနဲ့ အသုံးပြုနိုင်ပါတယ် ။

cp item1 item2

တခုထဲသော file သို့မဟုတ် directory ဖြစ်တဲ့ **item1** ကို file သို့မဟုတ် directory ဖြစ်တဲ့ **item2** ထဲကိုကူးထည့်ရန်အတွက် (အပေါ်က command ပုံစံကို) အသုံးပြုပြီးတော့

cp item... directory

file သို့မဟုတ် directory တွေအများကြီးကို directory တခုထဲကို copy ကူးထည့်ရန်အတွက် (အပေါ်က command ပုံစံကို) အသုံးပြုပါတယ် ။

Tables 4-4 နဲ့ 4-5 မှာ cp command အတွက် အသုံးများတဲ့ option (short option အပြင် သူ့အတိုင်း အလုပ်လုပ်ပေးတဲ့ long option တွေကိုပါ) list လုပ်ပြထားပါတယ် ။

Table 4-4: cp Options

Option	Meaning
-a, --archive	file တွေdirectory တွေနဲ့သူတို့ရဲ့ attribute တွေ ownership တွေနဲ့ permission တွေ အပါအဝင်အကုန်လုံးကို copy ကူးပေးပါတယ် ။ ပုံမှန်အားဖြင့်တော့ copy လုပ်တဲ့အခါမှာ copy လုပ်တဲ့ user ရဲ့ default attribute တွေကိုပဲယူပြီးလုပ်ပေးပါတယ် ။
-i, --interactive	ရှိပြီးသား file တခုအပေါ်မှာ overwriting မလုပ်ခင် user က confirmation လုပ်ဖို့ အတွက် prompt ပေါ်ပေးပါတယ် ။ တကယ်လို့ဒီ option ကိုမထည့်ထားခဲ့ရင် cp က silently (user ကိုဘာမှပြန်မမေးတော့ပဲ) file တွေကို တိတ်တဆိတ် overwrite လုပ်လိုက်မှာဖြစ်ပါတယ် ။
-r, --recursive	directory တွေနဲ့သူတို့ထဲက(ပါသမျှ) content တွေအကုန်လုံးကို Recursively copy လုပ်ပေးမှာဖြစ်ပါတယ် ။ directory တွေကို copy ကူးတဲ့အခါမှာ ဒီ option (သို့မဟုတ် -a

	option) ကိုထည့်ဖို့လိုအပ်ပါတယ် ။
-u, --update	file တွေကို directory တခုကနေနောက်ထပ် directory တခုဆီကို copy ကူးတဲ့အခါ မှာ copy ကူးပြီးသွားထည့်မယ့် destination directory ထဲကသက်ဆိုင်ရာ file တွေထက် version ပိုမြင့်နေတာ သို့မဟုတ် မရှိသေးတဲ့ file တွေကိုပဲကူးထည့်ပေးမှာဖြစ်ပါတယ် ။
-v, --verbose	copy လုပ်ပြီးတာနဲ့သတင်းပြန်ပို့တဲ့ (copy ကူးပြီးကြောင်း) message ကိုပြပေးမှာဖြစ်ပါတယ် ။

Table 4-5: cp Examples

Command	Results
cp file1 file2	file1 ကို file2 ထဲကူးထည့်ပေးမှာဖြစ်ပါတယ် ။ တကယ်လို့ file2 ကရှိပြီးသားဆိုရင် file1 ထဲက content တွေကို file2 ထဲကို overwrite လုပ်သွားမှာဖြစ်ပါတယ် ။ တကယ်လို့ file2 ကမရှိဘူးဆိုရင်သူက file2 ဆိုတဲ့နာမည်နဲ့ (အသစ်တခု)တည်ဆောက် (ပြီးတော့ file1 ထဲက content တွေကိုကူးထည့်) ပေးသွားမှာဖြစ်ပါတယ် ။
cp -i file1 file2	တကယ်လို့ file2 ကရှိပြီးသားဆိုရင် (overwrite လုပ်မှာလားဆိုတဲ့) user prompt ပေါ်လာ မှာကလွဲရင်ကျန်တာက အပေါ်ကအတိုင်းပါပဲ ဖြစ်ပါတယ် ။
cp file1 file2 dir1	file1 နဲ့ file2 ကို dir1 (ဆိုတဲ့ directory) ထဲကို copy ကူးပေးမှာဖြစ်ပါတယ် ။ dir1 က အစ ထဲကဆောက်ပြီးသားဖြစ်နေရပါမယ် ။
cp dir1/* dir2	wildcard တခုကိုသုံးပြီးတော့ dir1 ထဲက file တွေအကုန်လုံးကို dir2 ထဲကို copy လုပ်ပေး မှာဖြစ်ပါတယ် ။ dir2 က အစထဲကဆောက်ပြီးသားဖြစ်နေရပါမယ် ။
cp -r dir1 dir2	dir1 (နဲ့သူ့ထဲက content တွေ) ကို dir2 ဆီကို copy ကူးပေးမှာဖြစ်ပါတယ် ။ တကယ်လို့ dir2 မရှိသေးရင်(မဆောက်ရသေးရင်)သူကဆောက်ပေးမှာဖြစ်ပြီးတော့ dir1 ထဲက content တွေလိုပဲ (dir2 ထဲမှာ) အတူတူပဲဖြစ်နေပါလိမ့်မယ် ။

mv—Move and Rename Files

mv command က သူ့ကိုဘယ်လိုအသုံးပြုသလဲဆိုတဲ့အပေါ်မှာမူတည်ပြီးတော့ file နေရာရွှေ့တာ move နဲ့ file နာမည်ပြောင်းတာ rename ဂျ မျိုးစလုံးကိုလုပ်ပေးနိုင်ပါတယ် ။ (ဂျ မျိုးထဲက) ဘယ်ဟာကိုပဲဖြစ်ဖြစ်လုပ်ပြီးတဲ့ အခါမှာ file ရဲ့ နဂိုမူလနာမည်ကတော့ရှိတော့မှာမဟုတ်ပါဘူး ။ mv ကိုအသုံးပြုပုံဟာ cp နဲ့အတော်လေးဆင်တူပါ တယ် ။

mv item1 item2

file သို့မဟုတ် directory ဖြစ်တဲ့ item1 ကို item2 ထဲကို move သို့မဟုတ် rename လုပ်မယ် ။ ဒါမှမဟုတ်

mv item... directory

directory တခုကနေနောက်တခုဆီကို item တခု သို့မဟုတ် တခုထက်ပိုပြီးတော့ move လုပ်မယ် ။ Table 4-6 နဲ့ 4-7 မှာ ပြထားသလိုပဲ mv ဟာ cp နဲ့ option အများအပြား share မျှဝေသုံးစွဲ (သွားတူ) ထားပါတယ် ။

Table 4-6: mv Options

Option	Meaning
-i, --interactive	ရှိပြီးသား file ပေါ်မှာ overwrite မလုပ်ခင် user ရဲ့ confirmation တောင်းဖို့အတွက် prompt ပေါ်လာပါမယ် ။ ဒီ option ကိုထည့်မထားခဲ့ရင် mv က silently ဘာမှမမေးတော့ပဲ တိတ်တဆိတ် overwrite လုပ်သွားမှာဖြစ်ပါတယ် ။
-u, --update	file တွေကို directory တခုကနေနောက်တခုဆီကိုရွှေ့တဲ့အခါမှာ file တွေကိုသွားထားမယ့် destination directory ထဲကသက်ဆိုင်ရာ file တွေထက် version ပိုမြင့်နေတာ သို့မဟုတ် မရှိသေးတဲ့ file တွေကိုပဲရွှေ့ပေးမှာဖြစ်ပါတယ် ။
-v, --verbose	ရွှေ့ပြီးတာနဲ့သတင်းပြန်ပို့တဲ့ (ရွှေ့ပြီးကြောင်း) message ကိုပြပေးမှာဖြစ်ပါတယ် ။

Table 4-7: mv Examples

Command	Results
mv file1 file2	file1 ကနေ file2 ဆီကိုရွှေ့ပေးပါတယ် ။ တကယ်လို့ file2 ရှိနေပြီးသားဆိုရင် file1 ရဲ့ content တွေနဲ့အတူ (file2 ထဲ) ကို overwrite လုပ်ပေးပါတယ် ။ file2 မရှိသေးရင် တည်ဆောက်ပေးပါတယ် ။ ကိစ္စ ၂ ခုစလုံး (file2 တည်ဆောက်ထားတာရှိရှိ မရှိရှိ) file1 ကတော့ကျန်ရှိမှာမဟုတ်တော့ပါဘူး ။
mv -i file1 file2	တကယ်လို့ file2 ကရှိပြီးသားဆိုရင် (overwrite လုပ်မှာလားဆိုတဲ့) user prompt ပေါ်လာမှာကလွဲရင်ကျန်တာက အပေါ်ကအတိုင်းပါပဲ ဖြစ်ပါတယ် ။
mv file1 file2 dir1	file1 နဲ့ file2 ကို dir1 (ဆိုတဲ့ directory) ထဲကို ရွှေ့ပေးမှာဖြစ်ပါတယ် ။ dir1 က အစထဲက ဆောက်ပြီးသားဖြစ်နေရပါမယ် ။
mv dir1 dir2	dir1 (နဲ့သူ့ထဲက content တွေ) ကို dir2 ဆီကို ရွှေ့ပေးမှာဖြစ်ပါတယ် ။ တကယ်လို့ dir2 မ

	ရှိသေးရင်(မဆောက်ရသေးရင်)သူကဆောက်ပေးမှာဖြစ်ပြီးတော့ dir1 ထဲက content တွေကို dir2 ဆီကို ရွှေ့ပေးပြီးတော့ dir1 ကိုဖျက်ပြစ်လိုက်မှာဖြစ်ပါတယ်။ ။
--	---

rm—Remove Files and Directories

rm command က file သို့မဟုတ် directory တွေကို remove ရွှေ့တာ(delete ဖျက်တာ) အောက်ပါအတိုင်းလုပ်ပေးပါတယ်။ ။

rm item...

item ရဲ့နေရာမှာ file သို့မဟုတ် directory တွေရဲ့နာမည်ဖြစ်မှာဖြစ်ပါတယ်။ ။

BE CAREFUL WITH RM !

Linux လိုမျိုး Unix-like operating system တွေမှာ (ဖျက်ပြီးသားကိုမဖျက်တော့အောင်ပြန်လုပ်လို့ရတဲ့) undelete command မရှိပါဘူး။ ။ ခင်ဗျားက rm နဲ့တစ်ခုခုကိုဖျက်လိုက်ပြီဆိုတာနဲ့ အဲ့ဒါက ပြန်မရတော့ပါဘူး။ ။ Linux က ခင်ဗျားကို smart ဖြစ်ပြီး ကိုယ်ဘာလုပ်နေတယ်ဆိုတာကိုယ်သိတဲ့လူလို့ယူဆထားပါတယ်။ ။

wildcard တွေတစ်ချင်းစီအလိုက်ကိုဂရုစိုက်ပါ။ ။ ဒါကိုခေတ်ဟောင်းဥပမာတစ်ခုလို့သဘောထားလိုက်ပါ။ ။ ခင်ဗျားက directory တစ်ခုထဲက HTML file တွေချည်းပဲ(ရွေးပြီး)ဖျက်ချင်တယ်ဆိုပါစို့။ ။ ဒါကိုလုပ်ဖို့အတွက် ဒီလိုရိုက်ရပါမယ်။ ။

rm *.html

ဒါကအမှန်ပါပဲ။ ။ ဒါပေမယ့်ခင်ဗျားက မတော်တဆ * နဲ့ *.html ကြားထဲမှာ(အောက်ကလိုမျိုး) space ရိုက်ထည့်မိသွားရင်တော့

rm * .html

rm command က directory ထဲက file တွေအကုန်လုံးကိုဖျက်ပြစ်လိုက်ပြီးတော့ directory ထဲမှာဘာ file မှမရှိပါဘူးလို့ပြန်ပြောနေပါလိမ့်မယ်။ ။

Here is a useful tip: ခင်ဗျားအနေနဲ့ဘယ်အချိန်မှာပဲဖြစ်ဖြစ် rm နဲ့ wildcard ကိုတွဲသုံးတဲ့အခါမှာ (ခင်ဗျားစာရိုက်ထားတာတွေကိုသေသေချာချာစစ်ပြီးရင်) wildcard တွေကို ls နဲ့စစ်ပါ။ ။ ဒါကခင်ဗျားကို ဖျက်ပစ်မယ့် file တွေကိုမြင်ရစေပါတယ်။ ။ (စစ်လို့) ပြီးပြီဆိုရင်တော့ up arrow key နှိပ်ပြီး command ကိုပြန်ခေါ်ပြီးတော့ ls နဲ့ rm ကိုအစားထိုးလိုက်ရုံပါပဲ။ ။

Table 4-8 နဲ့ 4-9 တွေမှာ rm အတွက်အသုံးများတဲ့ option တချို့ကို list လုပ်ပြထားပါတယ်။

Table 4-8: rm Options

Option	Meaning
-i, --interactive	ရှိပြီးသား file တစ်ခုကိုမဖျက်ခင်မှာ user က confirmation လုပ်ပေးဖို့အတွက် prompt ပေါ်ပေးပါတယ်။ တကယ်လို့ ဒီ option ကိုထည့်မထားခဲ့ရင် rm က silently ဘာမှမမေးတော့ပဲ တိတ်တဆိတ် overwrite လုပ်သွားမှာဖြစ်ပါတယ်။
-r, --recursive	directory တွေကို recursive နဲ့ ဖျက်သွားမှာဖြစ်ပါတယ်။ ဆိုလိုတာက တကယ်လို့ ဖျက်မယ့် directory တစ်ခုမှာ subdirectory တွေရှိနေခဲ့မယ်ဆိုရင် သူတို့ကိုပါဖျက်သွားမှာဖြစ်ပါတယ်။ directory တစ်ခုကိုဖျက်တော့မယ်ဆိုရင် ဒီ option ကိုမဖြစ်မနေထည့်ပေးရမှာဖြစ်ပါတယ်။
-f, --force	file မရှိတာတွေဘာတွေကိုအရေးမထားတော့တဲ့အပြင် prompt လည်းမပြတော့ပါဘူး။ ဒါက --interactive option ကိုလည်း overwrite လုပ်သွားမှာဖြစ်ပါတယ်။
-v, --verbose	ဖျက်ပြီးတာနဲ့သတင်းပြန်ပို့တဲ့ (ဖျက်ပြီးကြောင်း) message ကိုပြပေးမှာဖြစ်ပါတယ်။

Table 4-9: rm Examples

Command	Results
rm file1	file1 ကို silently ဘာမှမမေးတော့ပဲ တိတ်တဆိတ်ဖျက်သွားမှာဖြစ်ပါတယ်။
rm -i file1	file1 ကိုမဖျက်ခင်မှာ user ရဲ့ confirmation တောင်းဖို့အတွက် prompt ပေါ်လာပါမယ်။
rm -r file1 dir1	file1 နဲ့ dir1 နဲ့ သူ့ထဲက content တွေကိုဖျက်သွားမှာဖြစ်ပါတယ်။
rm -rf file1 dir1	တကယ်လို့ file1 သို့မဟုတ် dir1 မရှိခဲ့ရင် rm က silently ဘာမှမမေးတော့ပဲ တိတ်တဆိတ် ဖျက်သွားမှာကလွဲရင်ကျန်တာအပေါ်ကအတိုင်းပါပဲ။

In—Create Links

ln command ကို hard သို့မဟုတ် symbolic link တွေတည်ဆောက်တဲ့နေရာမှာသုံးပါတယ်။ သူ့ကို အောက်ပါနည်း ၂ နည်းထဲက တနည်းနဲ့အသုံးပြုပါတယ်။

In file link

နဲ့ hard link တစ်ခုတည်ဆောက်ပြီးတော့

In -s item link

နဲ့ symbolic link တခုတည်ဆောက်ပြီးတော့ item နေရာမှာ file သို့မဟုတ် directory ရဲ့နာမည်ထည့်ရမှာ ဖြစ်ပါတယ်။ ။

Hard Links

Hard link တွေဟာ နဂိုမူလ Unix ရဲ့ link တွေတည်ဆောက်တဲ့နည်းလမ်းဖြစ်ပါတယ်။ ။ symbolic link တွေကတော့ခေတ်သစ်နည်းဖြစ်ပါတယ်။ ။ default အနေနဲ့ကတော့ file တိုင်းမှာ သူ့ရဲ့ filename ပေးဖို့အတွက် Hard link တခုစီပါဝင်ပြီးသားဖြစ်ပါတယ်။ ။ ကျွန်တော်တို့အနေနဲ့ Hard link တခုကိုတည်ဆောက်တဲ့အခါမှာ ကျွန်တော်တို့အနေနဲ့ file တခုအတွက်နောက်ထပ် directory တခုကိုပါတည်ဆောက်ပြီးသားဖြစ်သွားပါတယ်။ ။ Hard link တွေမှာအရေးကြီးတဲ့ကန့်သတ်ချက် ၂ ခုရှိပါတယ်။ ။

- Hard link တခုဟာ သူ့ကိုယ်ပိုင် file system ရဲ့ပြင်ပက file တခုကို reference မယူနိုင်ပါဘူး။ ။ ဒါကဘာကို ဆိုလိုတာလဲဆိုတော့ link တခုဟာသူ့ကိုယ်တိုင်ရှိမနေတဲ့ disk partition ထဲက file တခုကို reference လုပ် မပေးနိုင်ပါဘူး လို့ဆိုလိုပါတယ်။ ။
- Hard link တခုဟာ directory တခုကို reference လုပ်မပေးနိုင်ပါဘူး။ ။

Hard link တခုကို file ကိုယ်တိုင်ကနေခွဲခြားလို့မရနိုင်ပါဘူး။ ။ symbolic link တခုပါတဲ့ directory list တ ခုနဲ့မတူတာက Hard link တခုပါတဲ့ directory list တခုက link ရဲ့အထူးတလည်ညွှန်ပြချက်တွေကိုမပြပေးပါဘူး။ ။ Hard link တခုကိုဖျက်လိုက်တဲ့အခါ link ကိုဖယ်ရှားပစ်လိုက်ပေမယ့် file ကိုယ်တိုင်ရဲ့ content တွေကတော့ file မှာရှိသမျှ link တွေအကုန်လုံးကိုမဖျက်သမျှ ဆက်လက်တည်ရှိနေမှာဖြစ်ပါတယ်။ ။ (ဒါပဲလေ ၊ သူ့ရဲ့ space ကို deallocated မလုပ်ဘူး။)

Hard link တွေကိုသတိထားဖို့အထူးအရေးကြီးပါတယ်။ ။ ဘာလို့လဲဆိုတော့ ခင်ဗျားအနေနဲ့သူတို့ကို မကြာခဏတွေ့နေရမှာဖြစ်ပါတယ်။ ။ ဒါပေမယ့်ခေတ်သစ် လုပ်နည်းကိုင်နည်းကတော့ ကျွန်တော်တို့ဆက်ပြောမယ့် symbolic link တွေကိုပိုသုံးကြပါတယ်။ ။

Symbolic Links

symbolic link တွေဆိုတာ Hard link တွေရဲ့ကန့်သတ်ချက်တွေကိုကျော်လွန်နိုင်ဖို့အတွက် ဖန်တီးထား တာဖြစ်ပါတယ်။ ။ symbolic link တွေက file သို့မဟုတ် directory ကို reference ယူထားတဲ့ text pointer ပါတဲ့ အထူး file အမျိုးအစား တခုကိုတည်ဆောက်ပြီး အလုပ်လုပ်ပါတယ်။ ။ ဒီလိုနည်းလမ်းနဲ့သူတို့ဟာ Windows shortcut တခုလိုမျိုးလုပ်ဆောင်ပေးတယ်ဆိုပေမယ့်လည်း ဟုတ်ပါတယ် သူတို့ဟာ Windows feature တွေထက် နှစ်ပေါင်းများစွာစောပြီးထွက်ပေါ်လာတာဖြစ်ပါတယ်။ ;)

symbolic link တခုရဲ့ pointed လုပ်ခြင်းခံထားရတဲ့ file တခုနဲ့ symbolic link ကိုယ်တိုင်ဟာတခုဆီက နေတခုကိုကြီးကြီးမားမား ခွဲခြားလို့မရနိုင်ပါဘူး ။ ဥပမာ - ခင်ဗျားအနေနဲ့ symbolic link မှာတခုခုသွားရေးလိုက်တဲ့ အခါမှာ reference file မှာလည်း (ခင်ဗျားရေးလိုက်တာကို) သွားရေးလိုက်ပါတယ် ။ မည်သို့ပင်ဆိုစေကာမူ ခင်ဗျား က symbolic link ကိုဖျက်လိုက်တဲ့အခါမှာ link တခုထဲကိုဖျက်လိုက်ပြီးတော့ file ကိုတော့သွားမဖျက်ပါဘူး ။ တကယ်လို့ symbolic link ထက်အရင် file ကိုဖျက်လိုက်ရင် link ကဆက်ရှိနေမှာဖြစ်ပေမယ့် ဘယ်အရာကိုမှ point လုပ်နေမှာတော့မဟုတ်တော့ပါဘူး ။ အဲ့ဒါမျိုးကိစ္စမှာဆိုရင် link ကို broken ကျိုးပျက် နေတယ်လို့ပြောကြပါတယ် ။ implementation တွေအတော်များများမှာ ls command က broken link တွေကိုသူတို့တည်ရှိနေကြောင်း ထူးခြားတဲ့အရောင် ၊ အနီရောင်လိုမျိုးနဲ့ ဖော်ပြပေးနေမှာဖြစ်ပါတယ် ။

link တွေရဲ့သဘောတရားက ရှုပ်ထွေးတယ်လို့ထင်ရပေမယ့် ခဏတောင့်ထားပါဦး ။ ကျွန်တော်တို့ ဒါတွေ အားလုံးကိုလုပ်ကြည့်သွားမှာဖြစ်ပြီးတော့ ရှင်းသွား (နားလည်သွား) မယ်လို့မျှော်လင့်ရပါတယ် ။

Let's Build a Playground

ကျွန်တော်တို့အနေနဲ့တကယ့် file manipulation ပြုလုပ်မှာဖြစ်တာကြောင့် ကျွန်တော်တို့ရဲ့ file manipulation command တွေနဲ့ကစားဖို့အတွက် လုံခြုံစိတ်ချရတဲ့နေရာတခုတည်ဆောက်ကြပါစို့ ။ ပထမဦးဆုံး အနေနဲ့ကျွန်တော်တို့မှာအလုပ်လုပ်ဖို့ directory တခုလိုအပ်ပါတယ် ။ ကျွန်တော်တို့ရဲ့ home directory ထဲမှာအဲ့ဒါ တခုတည်ဆောက်ပြီးတော့သူ့ကို playground လို့နာမည်ပေးပါမယ် ။

Creating Directories

mkdir command ကို directory တွေတည်ဆောက်တဲ့နေရာမှာသုံးပါတယ် ။ ကျွန်တော်တို့ရဲ့ playground ဆိုတဲ့ directory ကိုတည်ဆောက်ဖို့အတွက် ကျွန်တော်တို့အနေနဲ့ပထမဦးဆုံးကျွန်တော်တို့ရဲ့ home directory ထဲမှာရှိနေတာသေချာအောင်အရင်လုပ်ပြီးတော့မှ directory အသစ်တခုကိုတည်ဆောက်ပါမယ် ။

```
jok3r@lucy:~$ cd
```

```
jok3r@lucy:~$ mkdir playground
```

playground ကိုနည်းနည်းလောက်ပိုပြီးစိတ်ဝင်စားစရာကောင်းစေဖို့အတွက် သူ့အထဲမှာ dir1 နဲ့ dir2 လို့ အမည်ရတဲ့ directory တချို့ကိုတည်ဆောက်လိုက်ပါမယ် ။ ဒါကိုလုပ်ဖို့အတွက် ကျွန်တော်တို့အနေနဲ့ကျွန်တော်တို့ရဲ့ current working directory ကို playground ထဲကိုရွှေ့လိုက်ပြီးတော့ နောက်ထပ် mkdir command ကို execute လုပ်ပါမယ် ။

```
jok3r@lucy:~$ cd playground/  
jok3r@lucy:~/playground$ mkdir dir1 dir2
```

mkdir command က directory ၂ ခုလုံးကို command တခုထဲနဲ့တည်ဆောက်ဖို့ခွင့်ပြုပေးတဲ့ argument တခုထက်ပိုပြီးလက်ခံပေးတာကိုသတိထားကြည့်ပါ။

Copying Files

ပြီးရင် ကျွန်တော်တို့ရဲ့ playground ထဲမှာ data တချို့ထည့်ကြစို့။ ကျွန်တော်တို့က ဒါကို file တခုကို copy လုပ်ခြင်းဖြင့်ပြုလုပ်မှာဖြစ်ပါတယ်။ cp command ကိုအသုံးပြုပြီးတော့ /etc directory ထဲက passwd file ကို

current working directory ထဲကို copy လုပ်မှာဖြစ်ပါတယ်။

```
jok3r@lucy:~/playground$ cp /etc/passwd .
```

current working directory အတွက် အတိုကောက်ကိုကျွန်တော်တို့ဘယ်လိုသုံးထားတယ်ဆိုတာကို သတိပြုကြည့်ပါ။ period အစက်ကလေးတစက် (passwd ရဲ့အနောက်မှာ) အနေနဲ့သုံးထားပါတယ်။ ဒါကြောင့်အခု ကျွန်တော်တို့က ls ရိုက်ထည့်လိုက်ရင်ကျွန်တော်တို့အနေနဲ့ကျွန်တော်တို့ရဲ့ file တွေကိုမြင်တွေ့ရမှာဖြစ်ပါတယ်။

```
jok3r@lucy:~/playground$ ls -l  
total 12  
drwxrwxr-x 2 jok3r jok3r 4096 Apr 27 00:02 dir1  
drwxrwxr-x 2 jok3r jok3r 4096 Apr 27 00:02 dir2  
-rw-r--r-- 1 jok3r jok3r 3003 Apr 27 00:03 passwd  
jok3r@lucy:~/playground$
```

အခု ၊ အပျော်ပေါ့ ၊ ကျွန်တော်တို့ -v (verbose) option နဲ့ copy ထပ်လုပ်ပြီးတော့ သူဘာလုပ်ပေးလဲဆိုတာကြည့်ကြစို့။

```
jok3r@lucy:~/playground$ cp -v /etc/passwd .
```

```
'/etc/passwd' -> './passwd'
```

cp command က copy နောက်တစ်ထပ်လုပ်ပေးပါတယ် ။ ဒါပေမယ့် ဒီအကြိမ်မှာတော့သူကဘာ operation လုပ်ပေးတယ်ဆိုတာကိုအသိပေးတဲ့ message ကိုပြပေးပါတယ် ။ တခုသတိထားရမှာက cp က ပထမ copy လုပ်ထားတဲ့ file (passwd file) ပေါ်ကို overwrite လုပ်သွားတာကို ဘာသတိပေးချက်မှမပေးသွားပါဘူး ။ ထပ်ပြီးတော့ ဒါက cp ကနေခင်ဗျားကို ခင်ဗျားကိုယ်ခင်ဗျားဘာလုပ်နေတယ်ဆိုတာကိုသိတယ်လို့ ယူဆပေးလိုက်တာဖြစ်ပါတယ် ။ သတိပေးချက် warning လိုချင်ရင်တော့ ကျွန်တော်တို့အနေနဲ့ -i (interactive) option ကိုထည့်ပေးရမှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~/playground$ cp -i /etc/passwd .  
cp: overwrite './passwd'?
```

y တလုံးရှိက်ထည့်လိုက်ပြီးတော့ prompt ကိုတုန့်ပြန်လိုက်ခြင်းအားဖြင့် file ကို overwrite လုပ်သွားစေမှာဖြစ်ပြီးတော့ အခြား character (ဥပမာ - n) တွေထည့်လိုက်ရင် cp က file ကိုဘာမှမလုပ်ပဲဒီအတိုင်းထားလိုက်စေမှာဖြစ်ပါတယ် ။

Moving and Renaming Files

အခု | passwd ဆိုတဲ့နာမည်က ဆော့ရတာသိပ်မကောင်းတဲ့အပြင် ဒါက playground (အစမ်းကစားကြည့်တဲ့နေရာ) လည်းဖြစ်တဲ့အတွက်သူ့ကိုတခြားနာမည်တခုခုကိုပြောင်းကြည့်ကြစို့ ။

```
jok3r@lucy:~/playground$ mv passwd fun
```

ပျော်စရာလေးတွေကိုမျှဝေတဲ့အနေနဲ့ကျွန်တော်တို့ရဲ့ နာမည်ပြောင်းထားတဲ့ file ကို directory တွေဆီကိုလျှောက်ပို့ကြည့်ပြီးတော့ (နဂိုနေရာမှာ) ပြန်ထားပါမယ် ။

```
jok3r@lucy:~/playground$ mv fun dir1
```

ပထမဆုံး dir1 ဆီကို ရွှေ့ပါမယ် ပြီးရင်

```
jok3r@lucy:~/playground$ mv dir1/fun dir2
```

dir1 ကနေ dir2 ဆီကို ရွှေ့ပါမယ် ပြီးရင်

```
jok3r@lucy:~/playground$ mv dir2/fun .
```

နောက်ဆုံးမှာတော့ current working directory ကိုပြန်ယူလာပါမယ် ။ ပြီးရင် directory တွေအပေါ် သက်ရောက်တဲ့ mv ရဲ့ effect တွေကိုကြည့်ပါမယ် ။ အရင်ဆုံးကျွန်တော်တို့ရဲ့ data file ကို dir1 ထဲကိုနောက်တ ကြိမ်ပြန်ရွှေ့ပါမယ် ။

```
jok3r@lucy:~/playground$ mv fun dir1
```

အဲ့ဒါလုပ်ပြီးတဲ့နောက်မှာ dir1 ကို dir2 ထဲကိုရွှေ့ပြီးတော့ (ရောက်မရောက်ကို) ls နဲ့ confirm လုပ်ပါမယ် ။

```
jok3r@lucy:~/playground$ mv dir1 dir2
jok3r@lucy:~/playground$ ls -l dir2
total 4
drwxrwxr-x 2 jok3r jok3r 4096 Apr 27 00:37 dir1
jok3r@lucy:~/playground$ ls -l dir2/dir1
total 4
-rw-r--r-- 1 jok3r jok3r 3003 Apr 27 00:03 fun
```

တခုမှတ်ထားရမှာက dir2 ကရှိနှင့်ပြီးသား (ဆောက်ထားပြီးသား)မို့လို့သာ mv က dir1 ကို dir2 ထဲကိုရွှေ့ ပေးတာဖြစ်ပါတယ် ။ တကယ်လို့ dir2 က ဆောက်ထားပြီးသားမရှိခဲ့ရင် mv က dir1 ကို dir2 လို့နာမည်ပြောင်း လိုက်မှာဖြစ်ပါတယ် ။ နောက်ဆုံးအနေနဲ့က အားလုံးကို (သူ့နေရာသူ) ပြန်ထားကြစို့ ။

```
jok3r@lucy:~/playground$ mv dir2/dir1 .
jok3r@lucy:~/playground$ mv dir1/fun .
```

Creating Hard Links

အခုကျွန်တော်တို့ link တချို့လုပ်ကြည့်ပါမယ် ။ hard link တွေကိုအရင်လုပ်ပါမယ် ။ ကျွန်တော်တို့အနေနဲ့ ကျွန်တော်တို့ရဲ့ data file လိုဟာမျိုးကို link တချို့လုပ်ကြည့်ပါမယ် ။

```
jok3r@lucy:~/playground$ ln fun fun-hard
jok3r@lucy:~/playground$ ln fun dir1/fun-hard
jok3r@lucy:~/playground$ ln fun dir2/fun-hard
```

အခုဆိုရင်ကျွန်တော်တို့အနေနဲ့ fun ဆိုတဲ့ file ရဲ့ instance တွေရသွားပါပြီ ။ ကျွန်တော်တို့ရဲ့ playground directory ထဲကိုတချက်လောက်သွားကြည့်ကြစို့ ။

```
jok3r@lucy:~/playground$ ls -l
total 16
drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 14:38 dir1
drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 14:37 dir2
-rw-r--r-- 4 jok3r jok3r 3003 Apr 28 14:39 fun
-rw-r--r-- 4 jok3r jok3r 3003 Apr 28 14:39 fun-hard
jok3r@lucy:~/playground$
```

ခင်ဗျားအနေနဲ့တခုသတိထားကြည့်ရမှာက fun နဲ့ fun-hard တို့ရဲ့ listing လုပ်ထားပေးတဲ့ဒုတိယလိုင်းမှာ 4 တလုံးစီရှိနေတာကိုပါပဲ ။ အဲ့ဒါက (fun ဆိုတဲ့ file အတွက်) အခုရှိနေတဲ့ hard link အရေအတွက်ဖြစ်ပါတယ် ။ file တခုမှာအမြဲတမ်းအနည်းဆုံး link တခုရှိနေမယ်ဆိုတာကိုခင်ဗျားမှတ်မိနေနမှာပါတာလို့လဲဆိုတော့ file name တွေကို link တခုနဲ့ဖန်တီးရလို့ပါပဲ ။ ဒါဆိုရင် fun နဲ့ fun-hard ဟာ တကယ်တော့ တူညီတဲ့ file တခုထဲဖြစ်တယ်ဆိုတာကိုကျွန်တော်တို့အနေနဲ့ဘယ်လိုသိနိုင်မလဲ ? ဒီကိစ္စမှာတော့ ls ကသိပ်အကူအညီမဖြစ်ပါဘူး ။ fun နဲ့ fun-hard ၂ ခုစလုံးဟာ တူညီတဲ့ (file) size ဖြစ်နေတဲ့အချိန်မှာ (field 5 ကိုကြည့်ပါ) ကျွန်တော်တို့ရဲ့ listing က သူတို့ ၂ file စလုံးဟာတူညီတဲ့ file ဖြစ်ကြောင်းဘယ်လိုမှ မပြသနိုင်ပါဘူး ။ ဒီပြဿနာကိုဖြေရှင်းဖို့အတွက်ဆိုရင်ကျွန်တော်တို့အနေနဲ့နည်းနည်းလောက် နက်နက်လေးတူးဆွရမှာဖြစ်ပါတယ် ။

hard link တွေအကြောင်းစဉ်းစားတဲ့အခါမှာ file တွေမှာ file ရဲ့ content တွေပါဝင်တဲ့ data ပိုင်းနဲ့ file name ကိုကိုင်တွယ်ထားတဲ့name ပိုင်းဆိုပြီးအပိုင်း ၂ ပိုင်းနဲ့ပြုလုပ်ထားတယ်လို့တွေးဆထားတာအကူအညီဖြစ်စေပါ

တယ် ။ ကျွန်တော်တို့က Hard link တွေကိုတည်ဆောက်လိုက်တဲ့အခါမှာ တကယ်တော့ ကျွန်တော်တို့က တူညီတဲ့ data အပိုင်းတွေကိုညွှန်ပြနေတဲ့ ထပ်တိုး name အပိုင်းတွေကိုတည်ဆောက်လိုက်တာပဲဖြစ်ပါတယ် ။ system က inode လို့ခေါ်တဲ့ name အပိုင်းတွေနဲ့ဆက်စပ်နေတဲ့ chain of disk block တခုကို assign လုပ်လိုက်တာပဲဖြစ်ပါတယ် ။ အဲဒါကြောင့် hard link တွေတခုစီတိုင်းဟာ file ရဲ့ content တွေပါဝင်တဲ့ တိကျတဲ့ inode တခုဆီကိုညွှန်းပြနေကြပါတယ် ။

ls command မှာဒီအချက်အလက်ကိုဖော်ပြပေးနိုင်တဲ့နည်းလမ်းတခုရှိပါတယ် ။-အဲနည်းလမ်းကတော့ -li option နဲ့ invoke လုပ်တာပါပဲ ။

```
jok3r@lucy:~/playground$ ls -li
total 16
9832483 drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 14:46 dir1
9839779 drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 14:46 dir2
9839780 -rw-r--r-- 4 jok3r jok3r 3003 Apr 28 14:39 fun
9839780 -rw-r--r-- 4 jok3r jok3r 3003 Apr 28 14:39 fun-hard
jok3r@lucy:~/playground$
```

အခု version ရဲ့ listing ထဲမှာဆိုရင်တော့ ပထမဦးဆုံး field ဟာ inode number တွေဖြစ်ပါတယ် ။ ပြီးတော့ကျွန်တော်တို့မြင်နေရတဲ့အတိုင်း fun နဲ့ fun-hard ၂ ခုလုံးဟာတူညီတဲ့ inode number တွေကိုမျှဝေသုံးစွဲထားကြတဲ့အတွက်ကြောင့်သူတို့ဟာ တူညီတဲ့ file တခု (အတူတူပဲ) ဖြစ်တာသေချာသွားပါတယ် ။

Creating Symbolic Links

Symbolic link တွေကို Hard link တွေရဲ့အားနည်းချက် ၂ ချက်ကိုကျော်လွန်နိုင်ဖို့အတွက်ဖန်တီးခဲ့တာဖြစ်ပါတယ် ။ Hard link တွေက physical device တွေကို span မလုပ်ပေးနိုင်ပါဘူး ။ ပြီးတော့ Hard link တွေက directory တွေကို reference လုပ်မပေးနိုင်ပါဘူး ။ file တွေကိုပဲလုပ်ပေးနိုင်ပါတယ် ။ Symbolic link တွေဟာ special file type တွေဖြစ်ပြီးတော့ သူ့မှာ target file သို့မဟုတ် directory (ကို point လုပ်ထားတဲ့) text pointer တခုပါဝင်ပါတယ် ။

symbolic link တွေကိုတည်ဆောက်တာဟာ Hard link တွေကိုတည်ဆောက်တာနဲ့ဆင်တူပါတယ် ။

```
jok3r@lucy:~/playground$ ln -s fun fun-sym
jok3r@lucy:~/playground$ ln -s ../fun dir1/fun-sym
jok3r@lucy:~/playground$ ln -s ../fun dir2/fun-sym
```


ပထမဆုံးဥပမာကတော့တော်တော်လေးဒဲ့တိုးဆန်ပါတယ် ။ ကျွန်တော်တို့အနေနဲ့ -s option ကိုသုံးပြီး တော့ hard link တခုထက်စာရင် symbolic link တခုကိုဖန်တီးလိုက်တာဖြစ်ပါတယ် ။ ဒါပေမယ့် နောက် ၂ ခု ကျ တော့ရော ? တခုမှတ်ထားရမှာက ကျွန်တော်တို့အနေနဲ့ symbolic link တခုကိုဖန်တီးလိုက်တဲ့အခါမှာ ကျွန်တော်တို့ က target file ဟာ symbolic link နဲ့ဘယ်လိုဆက်နွှယ်နေတယ်ဆိုတဲ့ text description (စာသားဖော်ပြချက်) တခု ကိုဖန်တီးလိုက်တာဖြစ်ပါတယ် ။ ls ရဲ့ output ကိုကြည့်လိုက်မယ်ဆိုရင်ပိုမို(ရှင်းလင်း)လွယ်ကူသွားပါလိမ့်မယ် ။

```
jok3r@lucy:~/playground$ ls -l dir1
total 4
-rw-r--r--  4 jok3r jok3r 3003 Apr 28 14:39 fun-hard
lrwxrwxrwx  1 jok3r jok3r   6 Apr 28 15:43 fun-sym -> ../fun
jok3r@lucy:~/playground$
```

dir1 ထဲက fun-sym အတွက် listing ပြနေတာက သူက symbolic link တခုဖြစ်ကြောင်း ထိပ်ဆုံး field (အရှေ့ဆုံး) မှာ l တလုံးနဲ့ ပြနေပြီး တကယ်တော့သူက ../fun ကို point လုပ်နေတာဖြစ်ပြီး အဲ့ဒါကလည်း မှန် ပါ တယ် ။ fun-sym ရှိနေတဲ့နေရာကနေဆက်နွှယ်ပြရမယ်ဆိုရင် fun ကသူ့အထက်က directory ထဲမှာရှိနေတာဖြစ်ပါ တယ် ။ symbolic link file ရဲ့ length က 6 ဖြစ်နေတာကိုလည်းပဲသတိပြုရပါမယ် ။ သူ pointing လုပ်နေတဲ့ file ရဲ့ length ထက်စာရင် ../fun ထဲက string တွေရဲ့ character အရေအတွက် (6) ကိုပြနေတာဖြစ်ပါတယ် ။

symbolic link တွေကိုဖန်တီးတည်ဆောက်တဲ့အခါမှာ ခင်ဗျားအနေနဲ့ absolute pathname တွေကိုအခု လိုအသုံးပြုနိုင်သလို

```
jok3r@lucy:~/playground$ ln -s /home/jok3r/playground/fun dir1/fun-sym
```

relative pathname တွေကိုလည်းကျွန်တော်တို့အစောပိုင်းဥပမာတွေမှာလုပ်ပြခဲ့သလိုမျိုး အသုံးပြုနိုင်ပါ တယ် ။ relative pathname တွေကိုသုံးပြီးတည်ဆောက်တာကပိုပြီးလုပ်ဖို့ကောင်းပါတယ်ဘာလို့လဲဆိုတော့ သူက symbolic link ရှိနေတဲ့ directory ကိုနာမည်ပြောင်းတာ သို့မဟုတ် link မပျက်သွားစေပဲနေရာရွှေ့တာတွေကို လုပ် ခွင့်ပြုလို့ပါပဲ ။

ပုံမှန် regular file တွေအတွက်နောက်တခုအနေနဲ့က symbolic link တွေဟာ directory ကိုလည်း reference လုပ်ပေးပါတယ် ။

```
jok3r@lucy:~/playground$ ln -s dir1 dir1-sym
jok3r@lucy:~/playground$ ls -l
total 16
drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 15:46 dir1
lrwxrwxrwx 1 jok3r jok3r    4 Apr 28 16:21 dir1-sym -> dir1
drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 15:44 dir2
-rw-r--r--  4 jok3r jok3r 3003 Apr 28 14:39 fun
-rw-r--r--  4 jok3r jok3r 3003 Apr 28 14:39 fun-hard
lrwxrwxrwx 1 jok3r jok3r    3 Apr 28 15:41 fun-sym -> fun
jok3r@lucy:~/playground$
```

Removing Files and Directories

ကျွန်တော်တို့အရှေ့ပိုင်းမှာသင်ကြားခဲ့ပြီဖြစ်တဲ့အတိုင်း rm command ကို file တွေနဲ့ directory တွေကို ဖျက်တဲ့နေရာမှာသုံးပါတယ် ။ ကျွန်တော်တို့ဟာ ကျွန်တော်တို့ရဲ့ playground ကိုအနည်းငယ်ရှင်းလင်းသွားအောင် လုပ်တဲ့နေရာမှာသူ့ကိုသုံးပါမယ် ။ ပထမဦးဆုံးအနေနဲ့ကျွန်တော်တို့ရဲ့ Hard link တွေထဲကတစ်ခုကိုဖျက်ပါမယ် ။

```
jok3r@lucy:~/playground$ rm fun-hard
jok3r@lucy:~/playground$ ls -l
total 12
drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 15:46 dir1
lrwxrwxrwx 1 jok3r jok3r    4 Apr 28 16:21 dir1-sym -> dir1
drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 15:44 dir2
-rw-r--r--  3 jok3r jok3r 3003 Apr 28 14:39 fun
lrwxrwxrwx 1 jok3r jok3r    3 Apr 28 15:41 fun-sym -> fun
jok3r@lucy:~/playground$
```

မျှော်လင့်ထားတဲ့အတိုင်းအလုပ်ဖြစ်ပါတယ် ။ fun-hard ဆိုတဲ့ file ကပျက်သွားပြီးတော့ fun အတွက် link count ကလည်း directory listing ထဲကဒုတိယ field မှာပြထားတဲ့အတိုင်း 4 ကနေ 3 ကိုလျော့သွားပါတယ် ။ နောက်ထပ်ကျွန်တော်တို့က fun ဆိုတဲ့ file ကိုဖျက်ပြီးတော့သူ့ဘာလုပ်သွားလဲဆိုတာကိုသိနိုင်ဖို့ အပျော်သဘောနဲ့ -i option ကိုထည့်မှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~/playground$ rm -i fun
rm: remove regular file 'fun'?
```

prompt မှာ y ကိုရိုက်ထည့်ပေးလိုက်ရင် file ကိုဖျက်ပေးသွားမှာဖြစ်ပါတယ်။ ဒါပေမယ့် အခု ls ရဲ့ output ကိုတချက်ကြည့်လိုက်ပါ။ fun-sym မှာဘာဖြစ်သွားလဲသတိထားကြည့်လိုက်ပါ။ symbolic link တခုဟာ မရှိတော့တဲ့ file တခုကိုညွှန်းနေပြီဆိုကထဲက သူက broken ဖြစ်သွားပါပြီ။ (broken link တွေကို အနီရောင်နဲ့ပြနေပါလိမ့်မယ်။)

```
jok3r@lucy:~/playground$ ls -l
total 8
drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 15:46 dir1
lrwxrwxrwx 1 jok3r jok3r    4 Apr 28 16:21 dir1-sym -> dir1
drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 15:44 dir2
lrwxrwxrwx 1 jok3r jok3r    3 Apr 28 15:41 fun-sym -> fun
jok3r@lucy:~/playground$
```

Linux distribution တော်တော်များများမှာ broken link တွေကိုပြနိုင်အောင် ls ကို configure လုပ်ထားကြပါတယ်။ Fedora box ပေါ်မှာဆိုရင် broken link တွေကိုအနီရောင်မှိတ်တုပ်မှိတ်တုပ်စာသားလေးနဲ့ပြပေးပါတယ်။ broken link တွေရှိနေတာဟာ သူကိုယ်တိုင်ကိုအန္တရာယ်မဖြစ်စေပေမယ့်ရှုပ်စေပါတယ်။ တကယ်လို့ကျွန်တော်တို့က broken link တခုကိုသုံးကြည့်မယ်ဆိုရင် ကျွန်တော်တို့ဒီလိုမြင်ရမှာဖြစ်ပါတယ်။

```
jok3r@lucy:~/playground$ less fun-sym
fun-sym: No such file or directory
jok3r@lucy:~/playground$
```

နည်းနည်းသန့်ရှင်းရေးလုပ်ကြစို့။ ကျွန်တော်တို့ symbolic link တွေကိုဖျက်ပါမယ်။

```
jok3r@lucy:~/playground$ rm fun-sym dir1-sym
jok3r@lucy:~/playground$ ls -l
total 8
drwxrwxr-x 2 jok3r jok3r 4096 Apr 28 15:46 dir1
```

symbolic link တွေပုံသင်္ကေတသတ်မှတ်ထားရမှာတခုက file operation တွေအများစုက link ရဲ့ target ပေါ်မှာလုပ်ဆောင်ကြတာဖြစ်ပြီးတော့ link ပေါ်မှာလုပ်နေတာမဟုတ်ဘူး ဆိုတဲ့အချက်ပါပဲ။ ဘာပဲဖြစ်ဖြစ် rm က တော့ချင်းချက်ပါ။ ခင်ဗျားက link တခုကိုဖျက်လိုက်တဲ့အခါ link ပဲပျက်သွားတာဖြစ်ပြီးတော့ target ကပျက်မသွားပါဘူး။

နောက်ဆုံးအနေနဲ့ကျွန်တော်တို့ကကျွန်တော်တို့ရဲ့ playground ကိုဖျက်ပါမယ်။ ဒါကိုလုပ်ဖို့အတွက် ကျွန်တော်တို့က ကျွန်တော်တို့ရဲ့ home directory ကိုပြန်သွားပြီးတော့ rm ကို recursive option (-r) နဲ့တွဲသုံးပြီး တော့ playground နဲ့ သူ့ရဲ့ content တွေ subdirectory တွေအပါအဝင် အကုန်လုံးကိုဖျက်ပါမယ်။

jok3r@lucy:~/playground\$ cd

jok3r@lucy:~\$ rm -r playground

jok3r@lucy:~\$

CREATING SYMLINKS WITH THE GUI

GNOME နဲ့ KDE ၊ ခုစလုံးရဲ့ file manager တွေမှာ symbolic link ဖန်တီးဖို့အတွက် လွယ်ကူပြီးအလိုအလျောက်ပြုလုပ်ပေးတဲ့ နည်းစနစ် easy and automatic method ပါဝင်ပါတယ်။ GNOME မှာဆိုရင် Ctrl နဲ့ Shift key ကိုဖိထားပြီး file ကို dragging ဆွဲလိုက်ရင် file ကို ကူးတာ copying (ရွှေ့တာ moving) တွေမလုပ်ပဲနဲ့ Link တခုတည်ဆောက်ပေးမှာဖြစ်ပါတယ်။ KDE မှာတော့ file ကျလာတဲ့အခါ menu အသေးလေးပေါ်လာပြီးတော့ copy လုပ်မှာလား move လုပ်မှာလား link ဆောက်မှာလား ဆိုပြီးရွေးပေးရပါတယ်။

Final Note

ဒီမှာကျွန်တော်တို့တော်တော်များများသင်လိုက်ရပြီးတော့ဒီအချက်အလက်တွေကိုသေသေချာချာနားလည်သဘောပေါက်ဖို့အတွက်ကတော့အချိန်ယူပြီးလေ့လာရမှာဖြစ်ပါတယ်။ playground exercise ကို နားလည်သဘောပေါက်လာသည်အထိ ထပ်ခါ ထပ်ခါ လေ့ကျင့်ပေးပါ။ အခြေခံ file manipulation command နဲ့ wildcard တွေကိုကောင်းကောင်းကြီးနားလည်ထားဖို့အရေးကြီးပါတယ်။ playground exercise ကိုပိုပြီးကျယ်ကျယ်ပြန့်ပြန့် လေ့ကျင့်နိုင်စေဖို့ file တွေ directory တွေကိုလွတ်လပ်စွာ (ကိုယ့်သဘောအတိုင်း) ထပ်ထည့်ပါ။ wildcard တွေကို အသုံးပြုပြီးတော့ file တွေကိုညွှန်းတာကို operation တွေမျိုးစုံမှာလုပ်ကြည့်ပါ။ link တွေပုံသင်္ကေတတဲ့ သဘောတရားတွေကအစပိုင်းမှာအနည်းငယ်ရှုပ်ထွေးနေနိုင်ပေမယ့် သူတို့ဘယ်လိုအလုပ်လုပ်သလဲဆိုတာကိုအချိန်ယူပြီးလေ့လာပါ။ သူတို့က (ခင်ဗျားအတွက်) အသက်ကယ်ဆေးလည်းဖြစ်လာနိုင်ပါတယ်။

(link တွေပုံသင်္ကေတပြီးသိသဘောပေါက်စေဖို့ youtube link ထည့်ပေးလိုက်ပါတယ် ။ ကြည့်ဖြစ်အောင်ကြည့်ပါ ။

Hard vs Soft Links in Linux (Linux Links)

<https://www.youtube.com/watch?v=4-vye3QFTFo>

MicroNuggets: Hard Links versus Soft Links Explained

https://www.youtube.com/watch?v=lW_V8oFxQgA

)

5

WORKING WITH COMMANDS

အခုအချိန်အထိတော့ကျွန်တော်တို့မှာ နားလည်ဖို့ခက်တဲ့ command အတွဲလိုက်တွေကိုပဲတွေ့နေရပြီး တော့သူတို့တစ်ခုမှာလည်း နားလည်ရခက်တဲ့ option တွေနဲ့ argument တွေပါဝင်နေပါတယ် ။ အခု Chapter မှာ ကျွန်တော်တို့ကအဲဒီ နားလည်ရခက်ခဲနေမှုတချို့ကိုပယ်ဖျက်နိုင်အောင်ကြိုးစားကြည့်မှာဖြစ်တဲ့အပြင်ကျွန်တော်တို့ ကိုယ်ပိုင် command တွေကိုတောင်ဖန်တီးကြည့်မှာဖြစ်ပါတယ် ။ အခု Chapter မှာမိတ်ဆက်ပေးမယ့် command တွေကတော့ ဒီဟာတွေပဲဖြစ်ပါတယ် ။

- **type** — command name တခုကိုဘယ်လို interpret လုပ်သလဲဆိုတာကိုညွှန်ပြပေးပါမယ် ။
- **which** — ဘယ် executable program က execute လုပ်မှာလဲဆိုတာကိုဖော်ပြပေးပါမယ် ။
- **man** — command တခုရဲ့ manual page ကိုဖော်ပြပေးပါမယ် ။
- **apropos** — appropriate command တွေရဲ့ list ကိုဖော်ပြပေးပါမယ် ။
- **info** — command တခုရဲ့ info entry ကိုဖော်ပြပေးပါမယ် ။
- **whatis** — command တခုရဲ့အကြမ်းဖျင်းအကျဉ်းချုပ်ဖော်ပြချက်ကိုဖော်ပြပေးပါမယ် ။
- **alias** — command တခုရဲ့ alias ဖန်တီးပေးပါမယ် ။

What Exactly Are Commands?

command တခုဟာ အောက်ဖော်ပြပါ ၄ ခုထဲက တခု ဖြစ်နိုင်ပါတယ် :

- **An executable program** /usr/bin ထဲမှာကျွန်တော်တို့မြင်ခဲ့တဲ့ file တွေလိုပါပဲ ။ ဒီအမျိုးအစားအတွင်း မှာဆိုရင် program တွေက binary တွေကို compile လုပ်ပေးနိုင်ပါတယ် ။ C နဲ့ C++ တွေနဲ့ရေးထားတဲ့

program လိုဟာမျိုးတွေကိုပေါ့ ၊ ဒါမှမဟုတ် scripting language တွေနဲ့ရေးထားတဲ့တွေ ၊ Shell , Perl, Python, Ruby စသည်ဖြင့် ။

- **A command built into the shell itself** bash က shell builtin လို့ခေါ်တဲ့ command တွေကို အတွင်းပိုင်းမှာ support ပေးထားပါတယ် ။ ဥပမာအနေနဲ့ cd command ဆိုတာလည်း shell builtin တခု ပါပဲ ။
- **A shell function** Shell function တွေဆိုတာ shell script အသေးစားလေးတွေဖြစ်ပြီးတော့ environment ထဲကိုပေါင်းစပ်ထားတာဖြစ်ပါတယ် ။ နောက်ပိုင်း Chapter တွေကျမှကျွန်တော်တို့က environment ကို configuring လုပ်တဲ့အကြောင်းနဲ့ shell function တွေရေးတဲ့အကြောင်းပြောမှာဖြစ်ပေမယ့်အခုတော့သူတို့ရှိတယ်ဆိုတာကိုသိထားပေးရုံပါပဲ ။
- **An alias** Alias တခုဆိုတာ command တခုဖြစ်ပြီးတော့ကျွန်တော်တို့ကိုယ်ပိုင်သတ်မှတ်ချက်ပြုလုပ် နိုင်ပြီး သူ့ကို တခြား command တွေကနေတည်ဆောက်ထားတာဖြစ်ပါတယ် ။

Identifying Commands

အဲ့ဒီ command လေးမျိုးထဲကဘယ်အမျိုးအစားကိုအသုံးပြုနေသလဲဆိုတာကိုအတိအကျသိနေတာက ခဏခဏအသုံးဝင်ပြီးတော့ Linux မှာအဲ့ဒါကိုရှာဖွေဖို့အတွက်နည်းလမ်းတွေတော်တော်များများဖြည့်ဆည်းပေးထားပါတယ် ။

type—Display a Command's Type

type command ဟာ shell builtin တခုဖြစ်ပြီးတော့ shell ကနေ execute လုပ်ပေးမယ့် command အမျိုးအစားကို ပေးထားတဲ့ (ခင်ဗျားသိချင်လို့ရှိက်ထည့်လိုက်တဲ့) command နာမည်နဲ့ (ရှာပြီးတော့) ဖော်ပြပေးမှာ ဖြစ်ပါတယ် ။ သူကဒီလိုအလုပ်လုပ်ပါတယ် ။

type command

command ဆိုတဲ့နေရာမှာခင်ဗျားသိချင်တဲ့ command ရဲ့နာမည်ထည့်ရမှာဖြစ်ပါတယ် ။ ဒီမှာ ဥပမာ တချို့ကိုဖော်ပြပေးလိုက်ပါတယ် ။

```
jok3r@lucy:~$ type type
type is a shell builtin
jok3r@lucy:~$ type ls
ls is aliased to `ls --color=auto'
cp is /usr/bin/cp
jok3r@lucy:~$
```

ဒီမှာ ကျွန်တော်တို့က မတူညီတဲ့ command ၃ ခုရဲ့ result ကိုမြင်နေရပါတယ် ။ သတိထားရမှာက ls command (Fedora system ကနေယူထားတာ) ဟာ တကယ်တော့ ls command မှာ --color=tty option ပေါင်း ထည့်ထားတဲ့ alias တခုပဲဖြစ်ပါတယ် ။ အခုဆိုရင်ကျွန်တော်တို့က ls ရဲ့ output မှာဘာလို့အရောင်တွေပါနေရလဲဆိုတာကိုသိသွားပါပြီ ။

which—Display an Executable’s Location

တခါတလေ system တခုပေါ်မှာ executable program တွေကို version တခုထက်ပိုပြီးတော့ သွင်းထားတတ်ပါတယ် ။ ဒါမျိုးက desktop system ပေါ်မှာပုံမှန်မဟုတ်ပေမယ့်လည်း server ကြီးတွေပေါ်မှာကျတော့လည်း ပုံမှန်မဟုတ်တာမျိုးမဟုတ်ပါဘူး ။ ပေးထားတဲ့ (ခင်ဗျားသိချင်လို့ရိုက်ထည့်လိုက်တဲ့) executable (program) ရှိနေတဲ့နေရာ location အတိအကျကိုဆုံးဖြတ်လိုတဲ့အခါမှာ which command ကိုအသုံးပြုပါတယ် ။

```
jok3r@lucy:~$ which ls
/usr/bin/ls
```

which က executable program တွေအတွက်သာအလုပ်လုပ်ပေးပြီးတော့တကယ့် executable program တွေကို အစားထိုးထားတဲ့ builtin တွေ သို့မဟုတ် alias တွေအတွက်အလုပ်မလုပ်ပေးပါဘူး ။ ကျွန်တော်တို့က shell builtin တခု (ဥပမာ - cd) ပေါ်မှာ which ကိုသုံးဖို့ကြိုးစားကြည့်တဲ့အခါမှာ ကျွန်တော်တို့အနေနဲ့ error message ပြတာ သို့မဟုတ် အလုပ်မလုပ်ပေးတာမျိုးပဲရမှာပါ ။

(ဒါက ubuntu 20.04 မှာစမ်းတာ ။ သူကအလုပ်မလုပ်ပေးတာ ။)

```
jok3r@lucy:~$ which cd
jok3r@lucy:~$
```

(ဒါက ubuntu အဟောင်းမှာစမ်းတာ ။ သူက error message ပြတာ ။)

```
jok3r@lucy:~$ which cd
/usr/bin/which: no cd in (/opt/jre1.6.0_03/bin:/usr/lib/qt-3.3/bin:/usr/kerberos/bin:/opt/jre1.6.0_03/bin:/usr/lib/ccache:/usr/local/bin:/usr/bin:/bin:/home/me/bin)
```

ဒါက “command not found” ဆိုတာကို fancy နည်းလမ်းလေးနဲ့ပြပေးတာပါပဲ ။

Getting a Command's Documentation

command တခုဟာဘာလဲဆိုတာကိုသိရင် ကျွန်တော်တို့အနေနဲ့အခု command အမျိုးအစားတခုစီရဲ့ Documentation တွေကို ရှာလို့ရပါပြီ။ ။

help—Get Help for Shell Builtins

bash မှာ shell builtin တွေတခုချင်းစီအတွက် built-in help facility တခုရှိပါတယ်။ ဒါကိုသုံးဖို့အတွက်ဆိုရင် help အနောက်မှာ shell builtin ရဲ့နာမည်ထည့်လိုက်ရုံပါပဲ။ ။ ဥပမာ -

```
jok3r@lucy:~$ help cd
```

```
cd: cd [-L|[-P [-e]] [-@]] [dir]
```

Change the shell working directory.

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

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

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

Options:

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

attributes as a directory containing the file attributes

The default is to follow symbolic links, as if `-L` were specified.

`..` is processed by removing the immediately previous pathname component back to a slash or the beginning of DIR.

Exit Status:

Returns 0 if the directory is changed, and if \$PWD is set successfully when

-P is used; non-zero otherwise.

jok3r@lucy:~\$

တချို့ program တွေက --help option ကို support မလုပ်ပေးပါဘူး။ ဒါပေမယ့် ကြိုးစားကြည့်လိုက်ပါ။ တခါတလေမှာသုက error message အနေနဲ့ result ထွက်လာတတ်ပြီးအဲဒါကတူညီတဲ့ usage information ဖြစ်နေတတ်ပါတယ်။

man—Display a Program's Manual Page

executable program တွေအများစုမှာ command line မှာသုံးဖို့အတွက်ရည်ရွယ်ပြီးတော့ထည့်ပေးထားတဲ့ manual သို့မဟုတ် man page လို့ခေါ်တဲ့ ပုံမှန် Documentation တခုရှိပါတယ်။ သူတို့ကိုကြည့်ဖို့အတွက် special paging program တခုဖြစ်တဲ့ man ကိုသုံးပါတယ်။ ဒီလိုမျိုးပေါ့

man program

program ရဲ့နေရာမှာကိုယ်ကြည့်ချင်တဲ့ command နာမည်ကိုထည့်ရမှာဖြစ်ပါတယ်။

man page မှာ format ပုံစံတွေအများကြီးရှိနေပေမယ့်လည်း ယျေဘုယျအားဖြင့်တော့သူ့မှာ command syntax ရဲ့ title တခု၊ synopsis တခု၊ command ရဲ့ ရည်ရွယ်ချက်ကိုဖော်ပြထားချက်တခုနဲ့ listing တခုနဲ့ command ရဲ့ option တွေတခုချင်းစီအတွက်ဖော်ပြထားချက်တွေပါဝင်ပါတယ်။ Man page တွေမှာ ဘယ်လိုပဲဖြစ်ဖြစ် ပုံမှန်အားဖြင့် example တွေပါဝင်ပါဘူး။ ပြီးတော့သူတို့ဟာ reference အနေနဲ့ပဲသုံးဖို့ရည်ရွယ်တာဖြစ်ပြီးတော့ tutorial တခုမဟုတ်ပါဘူး။ ဥပမာအနေနဲ့ ls command အတွက် man page မှာကြည့်ကြည့်ရအောင်။

jok3r@lucy:~\$ man ls

Linux system တွေအများစုမှာ man က manual page တွေကိုဖော်ပြဖို့အတွက် less ကိုအသုံးပြုတာကြောင့် ရင်းနှီးကျွမ်းကျင်ပြီးသား less command တွေက page ကိုဖော်ပြနေစဉ်မှာအသုံးဝင်ပါတယ်။

man ကနေဖော်ပြနေတဲ့ manual ဟာ section တွေအလိုက်ခွဲထားပြီးတော့ user command တွေအတွက်တင်မကပဲ system administration command တွေ programming interface တွေ file format တွေနဲ့အခြားအရာတွေအများကြီးအတွက်ကိုပါခြုံငုံမိစေပါတယ်။ Table 5-1 မှာ manual ရဲ့ layout ကိုဖော်ပြပေးထားပါတယ်။

Table 5-1: Man Page Organization

Section	Contents
1	User command တွေ
2	kernel system call တွေအတွက် Programming interface တွေ
3	C library ကိုသွားဖို့အတွက် Programming interface တွေ
4	device node တွေနဲ့ driver တွေအတွက် special file တွေ
5	File format တွေ
6	screensaver လိုမျိုး game နဲ့ ဖျော်ဖြေရေးတွေ
7	အမျိုးမျိုးသော သောင်းပြောင်းတွေ
8	System administration command တွေ

တခါတလေမှာကျွန်တော်တို့အနေနဲ့ကျွန်တော်တို့ရှာဖွေနေတဲ့အရာအတွက် manual ထဲက တိကျတဲ့ section တခုထဲမှာသွားရှာဖို့လိုအပ်ပါတယ်။ ပုံမှန်အားဖြင့် ဒါကမှန်ပါတယ်။ တကယ်လို့ ကျွန်တော်တို့က command နဲ့နာမည်တူနေတဲ့ file format တခုကိုရှာချင်တယ်ဆိုရင်ပေါ့။ တကယ်လို့ကျွန်တော်တို့က တိကျတဲ့ section နံပါတ်တခုကိုမရွေးချယ်ခဲ့ဘူးဆိုရင် ကျွန်တော်တို့အနေနဲ့ section 1 ထဲကပထမဦးဆုံးတူတဲ့ဟာကိုပဲရနိုင်ဖို့များပါတယ်။ section နံပါတ်တခုကိုအတိအကျရွေးပေးဖို့ဆိုရင်ကျွန်တော်တို့အနေနဲ့ man ကိုဒီလိုအသုံးပြုရမှာဖြစ်ပါတယ်။

man section search_term

ဥပမာ -

```
jok3r@lucy:~$ man 5 passwd
```

အဲ့လိုရိုက်ထည့်လိုက်ရင် man page က /etc/passwd file ထဲက file format ကိုဖော်ပြပေးမှာဖြစ်ပါတယ်။

```
jok3r@lucy:~$ apropos floppy
```

```
fdformat (8)      - low-level format a floppy disk
```

mbadblocks (1) - tests a floppy disk, and marks the bad blocks in the FAT
mformat (1) - add an MSDOS filesystem to a low-level formatted flopp...
mxtar (1) - Wrapper for using GNU tar directly from a floppy disk
jok3r@lucy:~\$

တလိုင်းချင်းစီရဲ့ output ရဲ့ ပထမဆုံး field က man page ရဲ့နာမည်တွေဖြစ်ပြီးတော့ ဒုတိယ field က (8 တွေ 1 တွေက) section တွေဖြစ်ပါတယ်။ တခုမှတ်ထားရမှာက man command မှာ -k option ထည့်လိုက်ရင် apropos command အတိုင်းအတိအကျအလုပ်လုပ်ပေးတယ်ဆိုတာကိုပါပဲ။

whatis—Display a Very Brief Description of a Command

whatis program က နာမည်နဲ့ (ခင်ဗျားရှာချင်တဲ့) keyword နဲ့အတိအကျတူတဲ့ man page ရဲ့ one-line description (တလိုင်းထဲနဲ့ပြီးတဲ့ဖော်ပြချက်အကျဉ်းချုပ်) ကိုဖော်ပြပေးပါတယ်။

jok3r@lucy:~\$ whatis ls
ls (1) - list directory contents
jok3r@lucy:~\$

THE MOST BRUTAL MAN PAGE OF THEM ALL

ကျွန်တော်တို့မြင်ခဲ့ရတဲ့အတိုင်း Linux နဲ့ အခြား Unix-like system တွေမှာပါတဲ့ manual page တွေက reference documentation အနေနဲ့ပဲရည်ရွယ်ထားတာဖြစ်ပြီးတော့ tutorial တွေအနေနဲ့မဟုတ်ပါဘူး။ man page တွေအတော်များများဟာဖတ်ရခက်ပါတယ် ဒါပေမယ့် ကျွန်တော့်အထင်တော့ အခက်ခဲဆုံးဆုပေးရမယ်ဆိုရင် bash အတွက်ရေးထားတဲ့ man page က ရမယ်ထင်ပါတယ်။ ကျွန်တော် ယခုစာအုပ်အတွက် research လုပ်စဉ် တုန်းက ကျွန်တော်က topic တော်တော်များများကိုခြုံငုံမိစေဖို့အတွက်သေသေချာချာကိုက်ရစိုက်ပြီးတော့ review လုပ်ခဲ့ပါတယ်။ print ထုတ်လိုက်တဲ့အခါမှာ စာမျက်နှာ ၈၀ ကျော်အောင်ရှည်ပြီးတော့ အလွန်အမင်းထူထဲနေပါတယ်။ ပြီးတော့ သူ့ရဲ့ structure ကလည်းလူသစ်တွေ (အခုမှ linux ကိုစ သုံးမယ့်သူတွေ) အတွက် လုံးဝကို အဓိပ္ပါယ်ဖော် (နားလည်)ဖို့ခက်ခဲနေစေပါတယ်။

အခြားတဖက်မှာလည်း သူကအလွန်ကိုတိကျပြီးကျစ်လစ်ပြည့်စုံ (လိုတိုရှင်း) ဖြစ်တဲ့အပြင်အလွန်အမင်းကို ပြည့်စုံလွန်းနေပါတယ်။ ဒါကြောင့်သတ္တိရှိရင်တော့ကိုယ့်ဘာသာကိုယ်စမ်းကြည့်လိုက် ပြီးတော့ ခင်ဗျားအနေနဲ့အဲ့ဒါ ကိုဖတ်နိုင်ပြီး သဘောပေါက်နားလည်လာမယ့်နေ့ကိုမျှော်လင့်နေပါတယ်။

info—Display a Program's Info Entry

GNU Project ကနေ man page အစား အသုံးပြုနိုင်ဖို့ info page တွေကိုထည့်ပေးထားပါတယ် ။ info page တွေဟာ named ဆိုတဲ့ reader program တခုနဲ့ဖော်ပြပေးပါတယ် ။ အတော်လေးအဆင်ပြေပါတယ် ။ info ဆိုတဲ့ info page တွေဟာ web page တွေလို hyperlink တွေဖြစ်ပါတယ် ။ ဒီမှာ ဥပမာ တခုပေးထားပါတယ် ။ (command line မှာ info ls လို့ရိုက်လိုက်ရင်ပေါ်လာတဲ့ဟာတွေကိုဥပမာပြထားပေးတာဖြစ်ပါတယ် ။)

Next: dir invocation, Up: Directory listing

10.1 'ls': List directory contents

=====

The 'ls' program lists information about files (of any type, including directories). Options and file arguments can be intermixed arbitrarily, as usual.

For non-option command-line arguments that are directories, by default 'ls' lists the contents of directories, not recursively, and omitting files with names beginning with '.'. For other non-option arguments, by default 'ls' lists just the file name. If no non-option argument is specified, 'ls' operates on the current directory, acting as if it had been invoked with a single argument of '.'.

By default, the output is sorted alphabetically, according to the locale settings in effect.(1) If standard output is a terminal, the output is in columns (sorted vertically) and control characters are output as question marks; otherwise, the output is listed one per line and control characters are output as-is.

-----Info: (coreutils)ls invocation, 57 lines --Top-----

info program က info file တွေကိုသွားဖတ်ပေးပါတယ် ။ အဲဒါတွေ (info file တွေ) က အကြောင်းအရာ topic တခုစီသာပါဝင်တဲ့ တခုစီသီးသန့်ရှိနေတဲ့ node တွေကိုသစ်ပင်ပုံစံမျိုး tree-structure အနေနဲ့ရှိနေကြတာဖြစ်ပါတယ် ။ info file တွေမှာ node တခုကနေနောက်တခုဆီကိုရွှေ့ပြောင်းသွားလို့ရတဲ့ hyperlink တွေပါဝင်ပါတယ် ။ hyperlink တခုကို သူ့ရဲ့အရှေ့ဆုံးက asterisk (*) လေးနဲ့ (hyperlink ဟုတ်မဟုတ်ဆိုတာကို) ဆုံးဖြတ်နိုင်ပြီးတော့ သူ့ကို activate လုပ်ဖို့အတွက်ကျတော့သူ့အပေါ် (asterisk အပေါ်) ကို cursor ချပြီး (asterisk အပေါ်မှာ double click နှိပ်ပြီး) Enter နှိပ်ရပါတယ် ။

info ကို invoke (run) ဖို့ဆိုရင် info လို့ရိုက်လိုက်ပြီးသူ့နောက်မှာ (space တချက်ခြားပြီးတော့ကိုယ်သိချင်တဲ့) program ရဲ့နာမည်ကိုရိုက်ထည့်ပေးရပါတယ် ။ Table 5-2 မှာ info page ပြနေစဉ် reader ကို control လုပ်လို့ရတဲ့ command တွေရဲ့ list ကိုပြထားပေးပါတယ် ။

(GNU info နဲ့ပက်သက်ပြီးနည်းနည်းပိုသိချင်ရင် ဒါလေးသွားဖတ်ကြည့်ပါ

<https://www.thegeeksearch.com/understanding-gnu-info-command-in-linux/>)

Table 5-2: info Commands

Command	Action
?	command help ကိုပြပေးမယ်
PAGE UP or BACKSPACE	previous page ကိုပြပေးမယ်
PAGE DOWN or Spacebar	next page ကိုပြပေးမယ်
n	Next ဖြစ်ပါတယ် next node ကိုပြပေးမယ်
p	Previous ဖြစ်ပါတယ် Previous node ကိုပြပေးမယ်
u	Up ဖြစ်ပါတယ် လက်ရှိပြနေတဲ့ node ရဲ့ parent node ဖြစ်ပါတယ် ။ သာမန်အားဖြင့်တော့ menu တခုပါပဲ ။
Enter	cursor ချထားတဲ့နေရာမှာရှိတဲ့ hyperlink နောက်ကိုလိုက်သွားပေးပါတယ် ။
q	(reader program ထဲက)ပြန်ထွက်သွားပါမယ် ။

အခုအချိန်အထိကျွန်တော်တို့ဆွေးနွေးခဲ့တဲ့ command-line program တော်တော်များများဟာ GNU Project ဖြစ်တဲ့ coreutils package ရဲ့အစိတ်အပိုင်းတွေဖြစ်ပါတယ် ။ ဒါကြောင့် ခင်ဗျားအနေနဲ့နောက်ထပ် အချက်အလက်တွေကိုအောက်ကလိုရိုက်ပြီးရှာဖွေတွေ့ရှိနိုင်ပါတယ် ။

```
jok3r@lucy:~$ info coreutils
```

အဲ့ဒါက coreutils package ကနေထည့်ပေးထားတဲ့ program တခုချင်းစီအတွက် documentation တွေ ကိုချိန်ဆက်ထားတဲ့ hyperlink တွေပါတဲ့ menu page တခုကိုဖော်ပြပေးမှာဖြစ်ပါတယ်။

README and Other Program Documentation Files

ခင်ဗျားရဲ့ system ပေါ်မှာ install လုပ်ထားတဲ့ software package တွေမှာ /usr/share/doc directory မှာ အခြေစိုက်ထားတဲ့ documentation file တွေပါရှိပါတယ်။ သူတို့အများစုဟာ plaintext format နဲ့သိမ်းထားတတ်ပြီးတော့သူတို့ကို less နဲ့ကြည့်လို့ရပါတယ်။ တချို့ file တွေက HTML format နဲ့ရှိနေတတ်ပြီးတော့သူတို့ကို web browser နဲ့ကြည့်လို့ရပါတယ်။ ကျွန်တော်တို့အနေနဲ့ .gz extension နဲ့အဆုံးသပ်ထားတဲ့ file တချို့နဲ့လည်းတွေ့ကြုံရနိုင်ပါတယ်။ အဲ့ဒါက သူ့ကို (file ကို) gzip compression program နဲ့ compress (ချုံ့) ထားတာဖြစ်တယ်လို့ဆိုလိုပါတယ်။ gzip package တွေမှာ zless လို့ခေါ်တဲ့ less ရဲ့ special version ပါဝင်ပြီးတော့သူက gzip - compress လုပ်ထားတဲ့ text file တွေထဲက content တွေကိုပြပေးမှာဖြစ်ပါတယ်။

Creating Your Own Commands with alias

အခု ကျွန်တော်တို့ရဲ့ ပထမဦးဆုံးသော programming အတွေ့အကြုံပေါ့ဗျာ ! ကျွန်တော်တို့ရဲ့ကိုယ်ပိုင် command တခုကို alias command အသုံးပြုပြီးတော့ပြုလုပ်ပါမယ်။ ဒါပေမယ့် ကျွန်တော်တို့မစခင်မှာ ကျွန်တော်တို့အနေနဲ့ command-line လှည့်ကွက်အသေးစားလေးတခုကိုအရင်ဆုံးဖော်ထုတ်ဖို့လိုအပ်ပါမယ်။ အဲ့ဒါကတော့ လိုင်းတလိုင်းထဲ (command တကြောင်းထဲ) မှာပဲ command တွေတခုထက်ပိုပြီးထည့်ဖို့အတွက် command တခုစီရဲ့ကြားမှာ semicolon character (;) တခုစီထည့်ပေးလိုက်ရင်ရတယ်ဆိုတာပါပဲ။ သူကဒီလိုမျိုးအလုပ်လုပ်ပါတယ်။

command1; command2; command3...

ကျွန်တော်တို့အသုံးပြုမယ့်ဥပမာက ဒီမှာပါ။

```
jok3r@lucy:~$ cd /usr;ls; cd -  
bin include lib32 libexec local share  
games lib lib64 libx32 sbin src  
/home/jok3r  
jok3r@lucy:~$
```

ကျွန်တော်တို့မြင်ခဲ့ရတဲ့အတိုင်းကျွန်တော်တို့က command ၃ ခုကို တလိုင်းထဲမှာပေါင်းထည့်ထားပါတယ်။ ပထမကျွန်တော်တို့က /usr ဆီကို directory change လိုက်ပါတယ်။ ပြီးတော့ directory ကို list လုပ်ပါတယ်။ ပြီးတော့နောက်ဆုံးအနေနဲ့ကျွန်တော်တို့က (cd - ကိုသုံးပြီးတော့) နဂိုမူလ directory ဆီကိုပြန်သွားလိုက်တဲ့အတွက်

ကျွန်တော်တို့စခဲ့တဲ့နေရာမှာပဲပြန်အဆုံးသပ်သွားခဲ့ပါတယ် ။ အခု ဒီ (command) အတွဲလိုက်ကြီးကို alias အသုံးပြုပြီးတော့ command အသစ်တခုအနေနဲ့ပြောင်းလိုက်ပါမယ် ။ ပထမဆုံးကျွန်တော်တို့လုပ်ရမှာကတော့ command အသစ်အတွက်နာမည်တခုကိုစိတ်ကူးယဉ်ကြည့်ဖို့ပါပဲ ။ test လို့ပေးကြည့်ကြစို့ ။ ကျွန်တော်တို့ဒါကိုမလုပ်ခင်မှာ test ဆိုတဲ့နာမည်က သုံးပြီးသားရှိမရှိစစ်ကြည့်တာပိုကောင်းမယ်ထင်ပါတယ် ။ ဒါကိုစစ်ကြည့်ဖို့အတွက် ကျွန်တော်တို့အနေနဲ့ type command ကိုနောက်တကြိမ်ထပ်သုံးနိုင်ပါတယ် ။

```
jok3r@lucy:~$ type test
test is a shell builtin
jok3r@lucy:~$
```

Oops! နာမည်ကယူပြီးသားဖြစ်နေပါတယ် ။ foo လို့ထားကြည့်ကြစို့ ။

```
jok3r@lucy:~$ type foo
bash: type: foo: not found
jok3r@lucy:~$
```

ကောင်းပြီ ။ foo ကယူသုံးထားတာမရှိသေးဘူး ။ ဒါကြောင့်ကျွန်တော်တို့ရဲ့ alias ကိုဖန်တီးကြစို့ ။

```
jok3r@lucy:~$ alias foo=='cd /usr; ls; cd -'
jok3r@lucy:~$
```

command ရဲ့ structure ကိုသတိထားကြည့်လိုက်ပါ ။

alias name='string'

alias ဆိုတဲ့ command အနောက်မှာ ကျွန်တော်တို့က alias ကိုနာမည်တခုပေးပြီးသူ့နောက်ကပ်လျက်မှာ (whitespace မခြားပဲနဲ့) ညီမျှခြင်း equal sign တခုထည့်လိုက်ပါမယ် ။ သူ့နောက်ကပ်လျက်က quoted string (' ဂုဏ်း) မှာ နာမည်ကို assign သွားလုပ်မယ့် (foo ဆိုတဲ့နာမည်ကို command line မှာရိုက်ထည့်လိုက်ရင်လုပ်ပေးမယ့်အလုပ်တွေ) အဓိပ္ပါယ် meaning တွေပါဝင်နေပါတယ် ။ ကျွန်တော်တို့ရဲ့ alias ကိုသတ်မှတ်လိုက်ပြီးတဲ့အခါမှာ shell က command တခုကိုစောင့်မျှော်နေတဲ့နေရာ (ဥပမာ - terminal ထဲမှာ) ဘယ်နေရာမှာမဆို သူ့ကိုအသုံးပြုလို့ရပါတယ် ။

```
jok3r@lucy:~$ foo
bin include lib32 libexec local share
```

```
games lib lib64 libx32 sbin src
/home/jok3r
jok3r@lucy:~$
```

ကျွန်တော်တို့အနေနဲ့ type command ကိုနောက်တစ်ထပ်သုံးပြီးတော့ကျွန်တော်တို့ရဲ့ alias ကိုကြည့်လို့ရပါတယ်။

```
jok3r@lucy:~$ type foo
foo is aliased to `cd /usr; ls; cd -`
jok3r@lucy:~$
```

ရှိပြီးသား command နာမည်တစ်ခုနဲ့မတူအောင်ကျွန်တော်တို့ကတစ်ရှောင်ပြီးကျွန်တော်တို့ရဲ့ alias ကို နာမည်ပေးခဲ့ပေမယ့်တစ်ခါတလေတော့တစ်ရှောင်မတူအောင်ပေးချင်တာမျိုးလည်းရှိနိုင်ပါတယ်။ ဒါမျိုးက တစ်ခါတလေမှာ အသုံးများတဲ့ command တစ်ခုချင်းစီမှာ သူ့ကို invocation လုပ်တဲ့အခါ အသုံးများတဲ့ option တွေထည့်ထားချင်တဲ့ အခါလုပ်လေ့ရှိပါတယ်။ ဥပမာအနေနဲ့ ls command ဟာ အရောင်တွေရရှိဖို့အတွက်ဘယ်လိုမကြာခဏ alias လုပ် တယ်ဆိုတာကို ကျွန်တော်တို့စောစောကမြင်ခဲ့တဲ့အတိုင်းပဲဖြစ်ပါတယ်။

```
jok3r@lucy:~$ alias ls
alias ls='ls --color=auto'
jok3r@lucy:~$
```

environment ထဲမှာရှိသမျှ alias တွေအကုန်လုံးကိုကြည့်ဖို့အတွက်ဆိုရင် alias command ကို argument မပါဘဲအသုံးပြုရမှာဖြစ်ပါတယ်။ အခုဒီမှာ Ubuntu 20.04 system မှာ default သတ်မှတ်ထားတဲ့ alias တချို့ကို ပြပေးလိုက်ပါတယ်။ သူတို့ဘာအလုပ်တွေလုပ်ပေးသလဲဆိုတာကိုအဖြေရှာကြည့်လိုက်ပါ။

```
jok3r@lucy:~$ alias
alias alert='notify-send --urgency=low -i "${[ $? = 0 ]} && echo terminal || echo error)" "${history|tail -
n1|sed -e \"s/^\s*[0-9]\+\s*//;s/[:&]\s*alert$/\"}'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias foo='cd /usr; ls; cd -'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
```



```
alias ll='ls -lF'
alias ls='ls --color=auto'
jok3r@lucy:~$
```

command-line ပေါ်မှာ alias သတ်မှတ်တဲ့အခါ ပြဿနာပိစီလေးတစ်ခုရှိနေပါတယ် ။ shell session ပြီးသွားတဲ့အခါ (terminal ကိုပိတ်လိုက်တဲ့အခါ) သူတို့ပျောက်သွားပါတယ် ။ နောက်ပိုင်း chapter တစ်ခုမှာကျွန်တော်တို့ရဲ့ alias တွေကို ကျွန်တော်တို့ log on လုပ်တဲ့အချိန်တိုင်းမှာ environment မှာ establish လုပ်ပေးတဲ့ file တွေမှာ ဘယ်လိုသွားပေါင်းထည့်မလဲဆိုတာကိုကြည့်ကြမှာဖြစ်ပါတယ် ။ ဒါပေမယ့် အခုအဖို့ကတော့ shell programming ကမ္ဘာထဲကိုသွားတဲ့ ကျွန်တော်တို့ရဲ့ ပထမဦးဆုံးသော ၊ သေးငယ်တယ်ဆိုပေမယ့်လည်း ၊ ခြေလှမ်း ကိုပဲပျော်ရွှင်ခံစားလိုက်ပါဦး ။

(alias အကြောင်းနည်းနည်းပိုသိအောင် ဒီမှာ သွားဖတ်နိုင်ပါတယ် ။ <https://fossbytes.com/alias-in-linux-how-to-use-create-permanent-aliases/>)

Revisiting Old Friends

အခုဆိုရင်ကျွန်တော်တို့က command တွေအတွက် documentation တွေဘယ်လိုရှာရမလဲဆိုတာကိုသိပြီဆိုတော့ အခုအထိကျွန်တော်တို့တွေ့လာခဲ့တဲ့ command တွေအကုန်လုံးအတွက် documentation တွေရှာကြည့်လိုက်ပါ ။ နောက်ထပ်ဘာ option တွေရှိနိုင်ဦးမလဲဆိုတာကိုလေ့လာပြီးတော့ စမ်းကြည့်လိုက်ပါ ။

REDIRECTION

အခု lesson မှာကျွန်တော်တို့က command-line ရဲ့အမိုက်ဆုံး feature လို့ဆိုနိုင်တဲ့အရာကိုလွှတ်ပေးမှာ ဖြစ်ပါတယ်။ I/O redirection ပဲဖြစ်ပါတယ်။ I/O ဆိုတာ input/output ကိုပြောတာဖြစ်ပြီးတော့ အဲဒီ facility နဲ့ ခင်ဗျားက input နဲ့ output ကို file တွေဆီကိုအသွားအပြန် redirect လုပ်နိုင်မှာဖြစ်သလို powerful command pipeline တွေလုပ်ဖို့အတွက်လည်း command တွေအများကြီးကိုဆက်သွယ်နိုင်မှာဖြစ်ပါတယ်။ ဒီ facility ကိုထုတ် ကြွားဖို့အတွက်ကျွန်တော်တို့ကအောက်ပါ command တွေကို မိတ်ဆက်ပေးမှာဖြစ်ပါတယ်။

- **cat** — file တွေကို Concatenate ချိတ်ဆက်ပေးမယ်။
- **sort** — text line တွေကိုစီပေးမယ်။
- **uniq** — repeat ဖြစ်နေတဲ့ line တွေကို report လုပ်တာ သို့မဟုတ် ချန်လှပ်ထားခဲ့ပေးမယ်။
- **wc** — line တခုချင်းစီအတွက် newline, word နဲ့ byte count တွေထုတ်ပြပေးမယ်။
- **grep** — pattern တခုနဲ့ကိုက်ညီတဲ့ line တွေထုတ်ပြပေးမယ်။
- **head** — file ရဲ့ပထမပိုင်းကိုထုတ်ပြပေးမယ်။
- **tail** — file ရဲ့နောက်ဆုံးပိုင်းကိုထုတ်ပြပေးမယ်။
- **tee** — standard input ကိုသွားဖတ်ပြီးတော့ standard output နဲ့ file တွေမှာသွားရေးပေးမယ်။

Standard Input, Output, and Error

ကျွန်တော်တို့အခုအချိန်အထိအသုံးပြုခဲ့တဲ့ program တွေအများစုဟာ output လိုဟာမျိုးတွေကိုထုတ်ပေး ပါတယ်။ ဒီ output ဟာတခါတလေမှာ ၂ မျိုး ရှိနေပါတယ်။ ပထမ ကျွန်တော်တို့မှာ program ရဲ့ result ရှိနေပါ တယ်။ ဒါပဲလေ။ program ကိုပုံစံချထားပြီးထုတ်လုပ်ခိုင်းထားတဲ့ data တွေပေါ့။ ဒုတိယ ကျွန်တော်တို့မှာ program ဘာဖြစ်နေတယ်ဆိုတာကိုကျွန်တော်တို့ကိုပြောပြနေတဲ့ status တွေနဲ့ error message တွေရှိနေပါတယ်။ ကျွန်တော်တို့က ls လို command မျိုးကိုကြည့်လိုက်မယ်ဆိုရင် သူကသူ့ရဲ့ status တွေနဲ့ error message တွေကို screen ပေါ်မှာပြနေတာကိုကျွန်တော်တို့မြင်နိုင်ပါတယ်။

ls လို အရာရာဟာ file တခုသာဖြစ်တယ် “everything is a file” ဆိုတဲ့ Unix Theme ကိုဦးထိပ်ထားတဲ့ program တွေမှာတော့သူတို့ရဲ့ result တွေကို standard output လို့ခေါ်တဲ့ (တခါတလေမှာ stdout လို့ဖော်ပြ တတ်တဲ့) special file တခုဆီကိုပို့တတ်ပြီးတော့ သူ့ရဲ့ message တွေကိုကျတော့နောက်ထပ် file တခုဖြစ်တဲ့ standard error (stderr) ဆီကိုပို့ပါတယ်။ default အနေနဲ့ကတော့ standard output နဲ့ standard error ၂ မျိုး စလုံးကို screen ဆီကိုချိတ်ဆက်ထားပြီးတော့ disk file တခုထဲကိုသိမ်းမထားပါဘူး။

နောက်တခုက program အများစုဟာ standard input (stdin) လို့ခေါ်တဲ့ facility တခုဆီကနေ input ယူ ပြီးတော့ အဲဒါကလည်း default အရ keyboard နဲ့ attached လုပ်ထားပါတယ်။

I/O redirection ကကျွန်တော်တို့ကို output တွေဘယ်ကိုသွားပြီးတော့ input တွေဘယ်ကလာရမလဲဆိုတာကို ပြောင်းလဲခွင့်ပေးထားပါတယ် ။ သာမန်အရတော့ output က screen ဆီကိုသွားပြီးတော့ input က Keyboard ဆီကနေလာပါတယ် ။ ဒါပေမယ့် I/O redirection ကအဲ့ဒါကိုပြောင်းလဲနိုင်ပါတယ် ။

Redirecting Standard Output

I/O redirection က standard output ဘယ်ကိုသွားမလဲဆိုတာအတွက် ကျွန်တော်တို့ကို ပြင်ဆင်သတ်မှတ်ခွင့်ပြုပါတယ် ။ standard output ကို screen ဆီကိုသွားမယ့်အစား နောက်ထပ် file တခုဆီကိုသွားစေချင်တဲ့အခါမှာ ကျွန်တော်တို့က redirection operator (>) ကိုအသုံးပြုပြီးသူ့အနောက်မှာ file ရဲ့နာမည်ကိုထည့်ပါတယ် ။ ဘာကြောင့်ကျွန်တော်တို့ကဒါကိုလုပ်ချင်ရမှာလဲ ? တခါတလေမှာ command တခုရဲ့ output ကို file တခုသွားသိမ်းထားတာအသုံးဝင်ပါတယ် ။ ဥပမာ - ကျွန်တော်တို့အနေနဲ့ ls command ရဲ့ output ကို screen ဆီကိုပို့မယ့်အစား ls-output.txt ဆိုတဲ့ file ဆီကိုပို့ဖို့ shell ကိုပြောလို့ရပါတယ် ။

```
jok3r@lucy:~$ ls -l /usr/bin directory/usr/bin > ls-output.txt
jok3r@lucy:~$
```

ဒီမှာ ကျွန်တော်တို့က /usr/bin directory ရဲ့ long listing ကို ဖန်တီးလိုက်ပြီးတော့သူ့ရဲ့ result ကို ls-output.txt ဆိုတဲ့ file ဆီကိုပို့လိုက်ပါတယ် ။ command ရဲ့ redirect လုပ်ထားတဲ့ output ကိုစစ်ဆေးကြည့်ကြစို့ ။

```
jok3r@lucy:~$ ls -l ls-output.txt
-rw-rw-r-- 1 jok3r jok3r 119567 May 4 16:08 ls-output.txt
jok3r@lucy:~$
```

ကောင်းတယ် ။ ကောင်းမွန်ကြီးမားတဲ့ file တခုပါပဲ ။ တကယ်လို့ကျွန်တော်တို့က file ကို less နဲ့ကြည့်လိုက်မယ်ဆိုရင် ls-output.txt ဆိုတဲ့ file ထဲမှာ ls command ရဲ့ result တွေပါဝင်နေတာကိုကျွန်တော်တို့မြင်ရပါလိမ့်မယ် ။

```
jok3r@lucy:~$ less ls-output.txt
```

အခု ကျွန်တော်တို့ရဲ့ redirection စမ်းသပ်မှုကိုနောက်တကြိမ်ထပ်လုပ်ကြည့်ပါမယ် ။ ဒါပေမယ့် အခုတကြိမ်မှာတော့လှည့်ကွက်လေးနဲ့လုပ်မှာပါ ။ ကျွန်တော်တို့က directory ရဲ့နာမည်ကိုတကယ်မရှိတဲ့ (မဆောက်ရသေးတဲ့ directory) နာမည်ကိုပေးမှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ ls -l /bin/usr > ls-output.txt
ls: cannot access '/bin/usr': No such file or directory
jok3r@lucy:~$
```

ကျွန်တော်တို့ error message တခုလက်ခံရရှိပါတယ်။ ဒါကအဓိပ္ပါယ်ရှိပါတယ်။ ဘာလို့လဲဆိုတော့ကျွန်တော်တို့က တကယ်မရှိတဲ့ directory ဖြစ်တဲ့ /bin/usr ကိုညွှန်းထားလို့ပါ။ ဒါပေမယ့် error message က ls-output.txt မှာသွားရေးရမယ့်အစားဘာဖြစ်လို့ screen ပေါ်မှာလာဖော်ပြနေတာလဲ ? အဖြေကတော့ ls program ကသူ့ရဲ့ error message ကို standard output ဆီကိုမပို့လို့ပါ။ အဲဒီအစား ကောင်းမွန်စွာရေးသားထားတဲ့ Unix program တွေအများစုအတိုင်း သူက သူ့ရဲ့ error message တွေကို standard error ဆီကိုပို့ပါတယ်။ ကျွန်တော်တို့က standard output ကိုပဲ redirect လုပ်ပြီးတော့ standard error ကိုမလုပ်ကထဲက error message က screen ဆီကိုဆက်ပို့နေပါတယ်။ ကျွန်တော်တို့ တမိနစ်လောက်နေရင် standard error ကိုဘယ်လို redirect လုပ်ရတယ်ဆိုတာကိုကြည့်ကြပါမယ်။ ဒါပေမယ့် ပထမအနေနဲ့ ကျွန်တော်တို့ရဲ့ output file ဘာဖြစ်သွားလဲဆိုတာကိုကြည့်ကြစို့။

```
jok3r@lucy:~$ ls -l ls-output.txt
-rw-rw-r-- 1 jok3r jok3r 0 May 4 16:34 ls-output.txt
jok3r@lucy:~$
```

file မှာအခုဆိုရင်သူ့ရဲ့ length(size) က zero ပဲရှိပါတော့တယ်။ ဒါက ဘာလို့လဲဆိုတော့ ကျွန်တော်တို့က output ကို redirect operator (>) နဲ့ redirect လုပ်တဲ့အခါမှာ (သွားရေးမယ့်)အဆုံးသပ် destination file က အစကနေပြန်စလို့ပါ။ ကျွန်တော်တို့ရဲ့ ls command က result မထုတ်ပေးပဲ error message ပဲထုတ်ပေးကထဲက redirection operation က file ကိုပြန်ရေးလိုက်ပြီးတော့ error message ကြောင့် ရပ်လိုက်တာဟာ သူ့ကိုယ်သူ အကုန်ရှင်းပြစ်လိုက်တာနဲ့အဆုံးသပ်သွားပါတယ်။ တကယ်တော့ကျွန်တော်တို့အနေနဲ့ file တခုကို တကယ့်ကို အကုန်ရှင်းပြစ်တဲ့အခါ (ဒါမှမဟုတ် file အလွတ်တခုအသစ်ဖန်တီးတဲ့အခါ) မှာကျွန်တော်တို့အနေနဲ့အောက်ပါလှည့်ကွက်ကိုအသုံးပြုလို့ရပါတယ်။

```
jok3r@lucy:~$ > ls-output.txt
jok3r@lucy:~$
```

redirection operator ရဲ့အရှေ့မှာ ရိုးရိုးလေးပဲ ဘာ command မှမထည့်ပဲ အသုံးပြုလိုက်ခြင်းဖြင့် file တခုကိုအကုန်ရှင်းပြစ်တာ ဒါမှမဟုတ် file အသစ်တခုကိုဖန်တီးပေးပါလိမ့်မယ်။ file အလွတ်ပေါ့။

ဒါဆိုရင်ကျွန်တော်တို့အနေနဲ့ file ကိုအစကထဲက overwrite လုပ်မယ့်အစား redirect လုပ်ထားတဲ့ output တွေကို file တခု(ရဲ့အဆုံးနားမှာ) ဘယ်လိုထပ်ခါထပ်ခါပေါင်းထည့်ကြမလဲ ? အဲ့ဒါမျိုးအတွက်ကျွန်တော်တို့က (>>) redirection operator ကို ဒီလိုအသုံးပြုပါတယ် ။

```
jok3r@lucy:~$ ls -l /usr/bin >> ls-output.txt
jok3r@lucy:~$
```

>> operator ကိုအသုံးပြုခြင်းဖြင့် result တွေဟာ output ထဲမှာ (file ရဲ့နောက်ဆုံးမှာ) ထပ်ခါထပ်ခါပေါင်းထည့်ပေးခြင်းဖြင့်အဆုံးသပ်သွားမှာဖြစ်ပါတယ် ။ တကယ်လို့ file ကတည်ဆောက်ပြီးသားရှိမနေခဲ့ရင် > operator ကိုအသုံးပြုသလိုပဲ (အသစ်တခု) တည်ဆောက်ပေးသွားမှာဖြစ်ပါတယ် ။ ဒါကိုစမ်းကြည့်ကြစို့ ။

```
jok3r@lucy:~$ ls -l /usr/bin >> ls-output.txt
jok3r@lucy:~$ ls -l /usr/bin >> ls-output.txt
jok3r@lucy:~$ ls -l /usr/bin >> ls-output.txt
jok3r@lucy:~$ ls -l ls-output.txt
-rw-rw-r-- 1 jok3r jok3r 503634 May 4 16:34 ls-output.txt
jok3r@lucy:~$
```

ကျွန်တော်တို့က command ကို ၃ ခါထပ်လုပ်လိုက်တဲ့အတွက် file size ကလည်း ၃ ဆလောက်ကြီးသွားပါတယ် ။ (ဆိုလိုတာက command တခါလုပ်တိုင်း file ရဲ့အဆုံးနားမှာကပ်ပြီးနောက်တခါ result တွေကိုထပ်ထည့်ပေးလို့ပါပဲ ။ ဒါကြောင့် ၃ခါလုပ်တော့ file size ကလည်း ၃ ဆလောက်ကြီးသွားတာပါ ။)

Redirecting Standard Error

Redirecting Standard Error မှာသူ့အတွက်အသုံးပြုရလွယ်ကူစေဖို့အတွက်သတ်မှတ်ထားတဲ့ redirecting operator မရှိပါဘူး ။ standard error ကို redirect လုပ်ဖို့အတွက်ကျွန်တော်တို့အနေနဲ့မဖြစ်မနေသူ့ရဲ့ file descriptor ဆီကို refer လုပ်ပေးရမှာဖြစ်ပါတယ် ။ program တခုက file stream တွေပေါ်ကဘယ်နေရာကိုမဆို output ကိုထုတ်လုပ်ပေးနိုင်ပါတယ် ။ ကျွန်တော်တို့အနေနဲ့အဲ့ဒီ file stream တွေထဲကပထမဆုံး ၃ ခုကို standard input (0), output(1) နဲ့ error(2) တွေအဖြစ် refer လုပ်နေတဲ့အချိန်မှာ shell ကသူတို့ကို အတွင်းပိုင်းကနေ file descriptor တွေ 0 ,1 နဲ့ 2 ဆိုပြီးအစဉ်လိုက် reference ယူထားပါတယ် ။ shell က descriptor number တွေကိုအသုံးပြုပြီးတော့ redirecting file တွေအတွက် notation တခုထုတ်ပေးပါတယ် ။ standard error က 2 ဖြစ်နေတဲ့အတွက်ကြောင့်ကျွန်တော်တို့အနေနဲ့ standard error ကိုဒီ notation နဲ့ redirect လုပ်ပေးနိုင်ပါတယ် ။

```
jok3r@lucy:~$ ls -l /bin/usr 2> ls-error.txt
jok3r@lucy:~$
```

ls-error.txt ဆိုတဲ့ file ဆီကို standard error တွေ redirection လုပ်ဖို့အတွက် file descriptor 2 ကို redirection operator ရှေ့ကပ်လျက်မှာထားထားပါတယ် ။

Redirecting Standard Output and Standard Error to One File

ကျွန်တော်တို့အနေနဲ့ command တွေအကုန်လုံးရဲ့ output တွေကို file တခုထဲမှာဖမ်းထား (သိမ်းထား) ချင်တဲ့ကိစ္စမျိုးတွေရှိလာတတ်ပါတယ် ။ ဒါကိုလုပ်ဖို့အတွက်ကျွန်တော်တို့က standard output နဲ့ standard error ကိုတချိန်ထဲမှာတပြိုင်နက် redirect လုပ်ပေးရမှာဖြစ်ပါတယ် ။ ဒါကိုလုပ်ဖို့အတွက် နည်းလမ်း ၂ ခုရှိပါတယ် ။ ပထမ နည်းကတော့ ဒီမှာ ရိုးရာနည်းလမ်းပါ ။ သူက shell version အဟောင်းတွေနဲ့အလုပ်လုပ်ပါတယ် ။

```
jok3r@lucy:~$ ls -l /bin/usr > ls-output.txt 2>&1
jok3r@lucy:~$
```

ဒီနည်းလမ်းကိုအသုံးပြုပြီးတော့ကျွန်တော်တို့က redirection ၂ ခုပြုလုပ်လိုက်ပါတယ် ။ ပထမ ကျွန်တော်တို့ standard output ကို ls-output.txt ဆိုတဲ့ file ဆီကို redirect ပြုလုပ်လိုက်ပါတယ် ။ ပြီးတော့ ကျွန်တော်တို့က file descriptor 2 (standard error) ကို file descriptor 1 (standard output) ဆီကို notation 2>&1 ကိုအသုံးပြုပြီးတော့ redirect ပြုလုပ်လိုက်ပါတယ် ။

Note: သတိပြုရမှာက redirection တွေကိုစီထားပုံကအရေးကြီးပါတယ် ။ standard output ကို redirecting လုပ်ပြီးတာနဲ့အမြဲတမ်း standard error ကမဖြစ်မနေပါလာရမှာပါ ။ ဒါမှ မဟုတ်ရင် အလုပ်လုပ်မှာမဟုတ်ပါဘူး ။ အပေါ်ကဥပမာထဲမှာ > ls-output.txt 2>&1 က standard error ကို ls-output.txt ဆိုတဲ့ file ဆီကို redirect လုပ်ပါတယ် ။ ဒါပေမယ့် တကယ်လို့ သူ့ကိုစီထားပုံက 2>&1 > ls-output.txt လို့ဖြစ်နေမယ်ဆိုရင် standard error က screen ဆီကို directed လုပ်သွား မှာဖြစ်ပါတယ် ။

လက်ရှိ bash ရဲ့ version မှာ ဒုတိယအနေနဲ့ ဒီလို (၂ခု)ပေါင်းပြီးတော့ redirection လုပ်တာတွေအတွက်ပိုပြီး streamlined ဖြစ်တဲ့နည်းလမ်းကိုထည့်ပေးထားပါတယ် ။

```
jok3r@lucy:~$ ls -l /bin/usr &> ls-output.txt
jok3r@lucy:~$
```

ဒီဥပမာထဲမှာ ကျွန်တော်တို့က single notation ဖြစ်တဲ့ &> ကိုအသုံးပြုပြီးတော့ standard output နဲ့ standard error ၂ ခုစလုံးကို ls-output.txt ဆီကို redirect လုပ်ထားပါတယ် ။

Disposing of Unwanted Output

တခါတလေမှာ တိတ်ဆိတ်ခြင်း က တကယ့်ရွှေပါပဲ ။ ပြီးတော့ကျွန်တော်တို့အနေနဲ့ command တခုရဲ့ output ကိုမလိုချင်ဘူး ။ ကျွန်တော်တို့ကအဲ့ဒါကိုလွှင့်ပစ်ချင်တယ် ။ ဒါက အထူးသဖြင့် error နဲ့ status message တွေကိုပြောတာဖြစ်ပါတယ် ။ system ကဒါကိုပြုလုပ်ဖို့အတွက်နည်းလမ်းတခုအနေနဲ့ /dev/null လို့ခေါ်တဲ့ special file တခုကိုထည့်ပေးထားပါတယ် ။ ဒီ file က bit bucket လို့ခေါ်တဲ့ system device တခုဖြစ်ပြီးတော့ သူက input တွေကိုလက်ခံပေးပြီးတော့ဘာမှပြန်မလုပ်ပေးပါဘူး ။ command တခုဆီကလာတဲ့ error message တွေကို ဖိနှိပ်ဖို့ အတွက်ကျွန်တော်တို့ကဒီဟာကိုလုပ်ပါတယ် ။

```
jok3r@lucy:~$ ls -l /bin/usr 2> /dev/null
jok3r@lucy:~$
```

/DEV/NULL IN UNIX CULTURE

bit bucket ဆိုတာ Unix ရဲ့ ရှေးဟောင်းအယူအဆတခုဖြစ်ပါတယ် ။ ပြီးတော့သူ့ရဲ့ စကြဝဠာ (အရာရာ) ဆန်နေမှုကြောင့် Unix ရိုးရာရဲ့အစိတ်အပိုင်းတော်တော်များများမှာသူကပေါ်ပေါက် (ပါဝင်) နေပါတယ် ။ ဒါကြောင့် တယောက်ယောက်ကခင်ဗျားရဲ့ command ကို dev null ထဲကိုပို့လိုက်တယ်လို့ပြောလာခဲ့မယ်ဆိုရင် သူ့ဘာကိုဆိုလို သလဲဆိုတာကိုခင်ဗျားသိတယ်မလား ။ နောက်ထပ် ဥပမာတွေကိုတော့ Wikipedia က article ဖြစ်တဲ့ <https://en.wikipedia.org/wiki/Devnull> မှာသွားဖတ်ကြည့်လိုက်ပါ ။

Redirecting Standard Input

အခုအချိန်အထိကျွန်တော်တို့အနေနဲ့ standard input ကနေအသုံးပြုနေတဲ့ command တခုမှမတွေ့ကြုံရ သေးတဲ့အတွက်ကြောင့် (တကယ်ကကျွန်တော်တို့တွေ့ခဲ့ပြီးပါပြီ ။ ဒါပေမယ့် ဒီ စပရိုက် ကိုခဏနေမှတိုက်ပါမယ်) ကျွန်တော်တို့အနေနဲ့ တခုလောက်မိတ်ဆက်ပေးဖို့လိုအပ်လာပါတယ် ။

cat—Concatenate Files

cat command က file တခု သို့မဟုတ် တခုထက်ပိုပြီးဖတ်နိုင်ပြီးတော့ standard output ဆီကို copy တွေ့ဒီလိုလုပ်ပေးပါတယ်။

cat [file...]

ကိစ္စတော်တော်များများမှာ ခင်ဗျားအနေနဲ့ cat command ကို DOS ထဲက type command ရဲ့ analogous အနေနဲ့ယူဆလို့ရပါတယ်။ ခင်ဗျားအနေနဲ့အဲဒါကို file တွေကို paging မပါဘဲဖော်ပြတဲ့နေရာမှာ အသုံးပြုနိုင်ပါတယ်။ ဥပမာ -

```
jok3r@lucy:~$ cat ls-output.txt
```

```
jok3r@lucy:~$
```

အဲဒါက ls-output.txt ထဲက content တွေကိုဖော်ပြပေးမှာဖြစ်ပါတယ်။ cat ကိုတခါတလေမှာ short text file တွေကိုဖော်ပြဖို့အတွက်အသုံးပြုပါတယ်။ cat က file တွေကိုတခုထက်ပိုပြီးတော့ argument အနေနဲ့ လက်ခံပေးနိုင်တာကြောင့် သူ့ကို file တွေကိုအတူတကွ (တခုထဲဖြစ်အောင်) join တဲ့နေရာမှာလည်းသုံးလို့ရပါတယ်။ ကျွန်တော်တို့က file အကြီးကြီးတခုကို download လုပ်ထားပြီးတော့သူ့ကိုအပိုင်းလေးတွေအများကြီးအဖြစ်ခွဲထား (multimedia file တွေကို Usenet ပေါ်မှာဒီလိုနည်းလမ်းနဲ့ခွဲခြမ်းထားလေ့ရှိပါတယ်။) ပြီးတော့ ကျွန်တော်တို့ကအဲဒါတွေကိုပြန်ပြီး join ချင်တယ် ဆိုပါစို့။ တကယ်လို့ file နာမည်တွေကလည်း -

movie.mpeg.001 movie.mpeg.002 ... movie.mpeg.099

ဆိုရင် ကျွန်တော်တို့အနေနဲ့ သူတို့ကို ဒီ command နဲ့ပြန်ပြီး join ပေးလို့ရပါတယ်။

```
jok3r@lucy:~$ cat movie.mpeg.0* > movie.mpeg
```

```
jok3r@lucy:~$
```

wildcard တွေက sorted order တွေထဲမှာအမြဲတမ်း expand လုပ်သွားတတ်တာကြောင့် argument တွေကိုမှန်မှန်ကန်ကန်စီထားတဲ့အတိုင်း arrange လုပ်ပေးသွားမှာဖြစ်ပါတယ်။

ဒါကအကုန်လုံးအဆင်ပြေကောင်းမွန်ပါတယ်။ ဒါပေမယ့် အဲဒါက standard input နဲ့ဘယ်လိုပက်သက်လို့လဲ ? အခုထိတော့မပက်သက်သေးပါဘူး။ တခြားတခုကိုစမ်းသပ်ကြည့်ကြစို့။ တကယ်လို့ကျွန်တော်တို့က cat ကို argument မပါဘဲနဲ့ထည့်လိုက်မယ်ဆိုရင်ဘာဖြစ်သွားမလဲ ?

```
jok3r@lucy:~$ cat movie.mpeg.0* > movie.mpeg
```

```
□
```

ဘာမှမဖြစ်သွားပါဘူး ။ သူကအဲ့ဒီနေရာမှာ hung နေတဲ့ပုံစံမျိုးနဲ့ (အဖြူရောင်လေးထောင့်အတုံးလေးမှိတ်တုပ်မှိတ်တုပ်ပြပြီးဘာမှမပြပဲနေပါလိမ့်မယ်) ထိုင်နေပါလိမ့်မယ် ။ သူ့ကိုကြည့်ရတာအဲ့ဒီလိုပုံစံမျိုးဖြစ်နေပါလိမ့်မယ် ။ ဒါပေမယ့်တကယ်တမ်းတော့သူက သူလုပ်ရမယ့်ဟာကိုအတိအကျလုပ်ပေးနေတာဖြစ်ပါတယ် ။

တကယ်လို့ cat ကို ဘာ argument မှမထည့်ပေးလိုက်ရင် သူက standard input ဆီကနေသွားဖတ်ပြီးတော့ standard input ကလည်း default အရ keyboard နဲ့ဆက်သွယ်ထားတဲ့အတွက်ကြောင့်သူကကျွန်တော်တို့ (keyboard) ကနေတစ်ခုခုရိုက်လာတာကိုစောင့်မျှော်နေတာဖြစ်ပါတယ် ။
ဒါကိုရိုက်ကြည့်လိုက်ပါ ။

```
jok3r@lucy:~$ cat movie.mpeg.0* > movie.mpeg
```

```
The quick brown fox jumped over the lazy dog.
```

ပြီးရင် CTRL -D ကိုရိုက်လိုက် (ဆိုလိုတာက CTRL key ကိုဖိထားပြီး D ကိုနှိပ်လိုက်ပါ) ပြီးတော့ standard input ပေါ်မှာအဲ့ဒါက end-of-file (EOF) ကိုရောက်သွားပြီဖြစ်တဲ့အကြောင်း cat ကိုပြောလိုက်ပါ ။

```
jok3r@lucy:~$ cat movie.mpeg.0* > movie.mpeg
```

```
The quick brown fox jumped over the lazy dog.
```

```
The quick brown fox jumped over the lazy dog.
```

filename ရဲ့ argument တွေမပါတဲ့အတွက်ကြောင့် cat က standard input ကို standard output ဆီကို copy ကူးပေးလိုက်ပါတယ် ။ ဒါကြောင့်ကျွန်တော်တို့ရဲ့ text line တွေက (၂ခါ) ထပ်နေတာဖြစ်ပါတယ် ။ ကျွန်တော်တို့အနေနဲ့ဒီလိုအမူအကျင့်ကို (လုပ်ဆောင်ပေးမှုကို) short text file တွေဖန်တီးတဲ့နေရာမှာအသုံးပြုနိုင်ပါတယ် ။ ကျွန်တော်တို့က အခု ဥပမာထဲကပါတဲ့စာသားတွေပါဝင်တဲ့ lazy_dog.txt လို့ခေါ်တဲ့ file တစ်ခုကိုဖန်တီးချင်တယ်ဆိုပါစို့ ။ ကျွန်တော်တို့ကဒီလိုမျိုးလုပ်မှာပါ ။

```
jok3r@lucy:~$ cat > lazy_dog.txt
```

```
The quick brown fox jumped over the lazy dog.
```

command ရဲ့နောက်မှာကျွန်တော်တို့က file ထဲကိုသွားထည့်ချင်တဲ့ text တွေကိုရိုက်ထည့်ပါ။ ။ ပြီးသွားရင် CTRL -D နှိပ် (ပြီးထွက်) ဖို့ကိုလည်းသတိရပါ။ ။ command line ကိုအသုံးပြုပြီးတော့ ကျွန်တော်တို့က ကမ္ဘာ့အတုံးဆုံး word processor ကိုအသုံးပြုရမယ် ! ကျွန်တော်တို့ရဲ့ result တွေကိုကြည့်ဖို့အတွက် ကျွန်တော်တို့အနေနဲ့ cat ကိုအသုံးပြုပြီးတော့ file ကို standard output ဆီကိုနောက်ထပ် copy ထပ်လုပ်ပါမယ်။ ။

```
jok3r@lucy:~$ cat lazy_dog.txt
```

```
The quick brown fox jumped over the lazy dog.
```

```
jok3r@lucy:~$
```

အခုဆိုရင် cat က standard input ကို file name argument ကိုပါထပ်ပေါင်းလက်ခံပေးတယ်ဆိုတာကို ကျွန်တော်တို့သိသွားပါပြီ။ ။ standard input ကို redirect လုပ်ကြည့်ကြစို့။ ။

```
jok3r@lucy:~$ cat < lazy_dog.txt
```

```
The quick brown fox jumped over the lazy dog .
```

```
jok3r@lucy:~$
```

< redirection operator ကိုအသုံးပြုပြီးတော့ကျွန်တော်တို့က standard input ရဲ့ source ကို keyboard ကနေ lazy_dog.txt ဆိုတဲ့ file ဆီကိုပြောင်းလိုက်ပါတယ်။ ။ result က single filename argument တခုကို passing လုပ်တာနဲ့အတူတူပဲဖြစ်နေတာကိုကျွန်တော်တို့မြင်ရမှာဖြစ်ပါတယ်။ ။ ဒါမျိုးက အထူးသဖြင့် single filename argument တခုကို passing လုပ်တာနဲ့ယှဉ်လိုက်ရင်အသုံးမဝင်ပါဘူး။ ဒါပေမယ့် သူ့ကို standard input ရဲ့ source တခုကို file အနေနဲ့သုံးပြီး လက်တွေ့ဥပမာလုပ်ပြတဲ့အနေနဲ့လုပ်ဆောင်တာဖြစ်ပါတယ်။ ။ တခြား command တွေက ကျွန်တော်တို့မကြာခင်မှာတွေ့ရမယ့်အတိုင်း standard input ပိုင်းမှာပိုကောင်းအောင်လုပ်ဆောင်ပေးနိုင်ကြပါတယ်။ ။

ကျွန်တော်တို့ရှေ့ဆက်မသွားခင်မှာ cat ရဲ့ man page ကိုတချက်လောက်လေ့လာကြည့်လိုက်ပါ။ ။ သူ့မှာ စိတ်ဝင်စားစရာ option တွေအများအပြားရှိနေပါတယ်။ ။

Pipelines

standard input ကနေ data တွေကို ဖတ်ပြီးတော့ standard output ဆီကိုပို့နိုင်တဲ့ command တွေရဲ့ စွမ်းအားကို pipeline လို့ခေါ်တဲ့ shell feature တခုကအသုံးပြုပါတယ်။ pipe operator (|) (မျဉ်းအထောင်ပုံစံ လေး) ကိုအသုံးပြုပြီးတော့ command တခုရဲ့ standard output ကို pipe လုပ်ပြီးတော့အခြား (command တခု ရဲ့) standard input ထဲကိုသွားထည့်လို့ရပါတယ်။

command1 | command2

ဒါကိုအပြည့်အစုံလက်တွေ့လုပ်ပြဖို့အတွက်ကျွန်တော်တို့မှာ command အချို့လိုအပ်ပါမယ်။ အဲဒီမှာ ကျွန်တော်တို့ သိပြီးသားတခု၊ သူက standard input ကိုလက်ခံပေးတယ်။ အဲဒီအကြောင်းဘယ်လိုပြောခဲ့လဲမှတ်မိ မလားမသိဘူး ? အဲဒါ less ပါ။ ကျွန်တော်တို့အနေနဲ့ less ကိုသုံးပြီးတော့ ဘယ် command မျိုးရဲ့ output ဖြစ်ဖြစ် သူ့ရဲ့ result ကို standard output ဆီပို့လိုက်တာတွေကို page by page တမျက်နှာချင်းစီဖော်ပြလို့ရပါတယ်။

```
jok3r@lucy:~$ ls -l /usr/bin | less
```

ဒါကအလွန်အမင်းကိုအသုံးဝင်ပါတယ်။ ဒီနည်းစနစ်ကိုအသုံးပြုပြီးတော့ကျွန်တော်တို့အနေနဲ့ standard output ထုတ်ပေးတဲ့ ဘယ်လို command မျိုးမဆိုရဲ့ output ကိုအလွယ်တကူ စစ်ဆေးနိုင်ပါတယ်။

Filters

Pipeline တွေကို မကြာခဏဆိုသလို data နဲ့ပက်သက်ပြီး ရှုပ်ထွေးလှတဲ့ operation တွေလုပ်ကိုင်ရာမှာ အသုံးပြုလေ့ရှိပါတယ်။ Pipeline တွေထဲမှာ command အများအပြားကိုအတူတကွစုပြီးထည့်လို့ရပါတယ်။ မကြာခဏဆိုသလိုပဲ command တွေကိုဒီလိုနည်းလမ်းနဲ့လုပ်တာကို filter လုပ်တယ်လို့ရည်ညွှန်းကြပါတယ်။ filter တွေက input ကိုလက်ခံမယ်၊ တနည်းနည်းနဲ့ပြောင်းလိုက်ပြီးတော့ output ပြန်ထုတ်ပေးပါတယ်။ ကျွန်တော် တို့ပထမဆုံးစမ်းကြည့်မယ့်ဟာက sort ဖြစ်ပါတယ်။ ကျွန်တော်တို့အနေနဲ့ /bin နဲ့ /usr/bin ထဲကရှိသမျှ executable program တွေအကုန်ပေါင်းထားတဲ့ list တခုလုပ်မယ်။ သူတို့ကို sorted order (အစီအစဉ်အတိုင်းဖြစ် အောင်) စီထားမယ် ပြီးရင် list ကိုပြန်ကြည့်မယ် လို့စိတ်ကူးယဉ်ကြည့်လိုက်ပါ။

```
jok3r@lucy:~$ ls /bin /usr/bin | sort | less
```

ကျွန်တော်တို့က directory ၂ ခု (/bin နဲ့ /usr/bin) ကိုသတ်မှတ်ပေးလိုက်တဲ့အတွက်ကြောင့် ls ရဲ့ output မှာလည်း directory ၂ ခုအတွက် (list) တခုစီအဖြစ် ၂ ပိုင်းကွဲသွားပါမယ်။ ကျွန်တော်တို့ရဲ့ pipeline ထဲမှာ sort ကိုထည့်လိုက်တဲ့အတွက် ကျွန်တော်တို့က data ကို sorted list တခုထဲအဖြစ် ထုတ်လုပ်ပေးဖို့ပြောင်းလဲလိုက်တာ ဖြစ်ပါတယ်။

uniq—Report or Omit Repeated Lines

uniq command ကိုတခါတလေမှာ sort နဲ့တွဲသုံးလေ့ရှိပါတယ်။ uniq က standard input သို့မဟုတ် single filename argument တခုဆီကနေ sorted list တခုလုပ်ထားတဲ့ data ကိုလက်ခံပေး (အသေးစိတ် အချက်အလက်အတွက် uniq ရဲ့ man page ကိုကြည့်ပါ) ပြီးတော့ default အရ list ထဲမှာဘယ်လို duplicate ဖြစ် နေတာတွေကိုမဆို (နာမည်တူပြီးတကြိမ်ထက်ပိုပါဝင်နေသော) ဖယ်ရှားပစ်မှာဖြစ်ပါတယ်။ ဒါကြောင့်ကျွန်တော် တို့ရဲ့ list မှာ duplicate ဖြစ်နေတာတွေမပါတာသေချာအောင် (ဒါပဲလေ ၊ နာမည်တူပြီး /bin နဲ့ /usr/bin directory ၂ ခုစလုံးမှာပါနေတဲ့ ဘယ် program ကိုမဆို) ကျွန်တော်တို့ရဲ့ pipeline မှာ uniq ကိုထည့်သွင်းမှာဖြစ်ပါ တယ်။

ဒီဥပမာထဲမှာ ကျွန်တော်တို့က uniq ကိုအသုံးပြုပြီးတော့ sort command ရဲ့ output ထဲက duplicate ဖြစ်နေတဲ့ဟာမှန်သမျှဖယ်ရှားပစ်မှာဖြစ်ပါတယ်။

```
jok3r@lucy:~$ ls /bin /usr/bin | sort | uniq -d | less
```

wc—Print Line, Word, and Byte Counts

wc (word count) command ကိုအသုံးပြုပြီးတော့ file တွေထဲက line တွေ word တွေနဲ့ byte တွေကို ဖော်ပြမှာဖြစ်ပါတယ်။ ဥပမာ -

```
jok3r@lucy:~$ wc ls-output.txt
1892 17861 119567 ls-output.txt
jok3r@lucy:~$
```

အခုကိစ္စမှာ သူက ls-output.txt ဆိုတဲ့ file ထဲမှာရှိတဲ့ line တွေ word တွေနဲ့ byte တွေရဲ့ နံပါတ် ၃ ခုကို print out ထုတ်ပြနေတာဖြစ်ပါတယ်။ ။ ကျွန်တော်တို့ရဲ့ အရင်က command တွေလိုပဲ တကယ်လို့ command-line argument တွေမပါဘဲ execute လုပ်ခဲ့မယ်ဆိုရင် wc က standard input ကိုလက်ခံပေးပါတယ်။ ။ -l option ကသူ့ရဲ့ output ကို line တွေချဉ်းပဲပြဖို့ကန့်သတ်ထားပါတယ်။ ။ ကျွန်တော်တို့ရဲ့ sorted list ထဲမှာရှိနေတဲ့ item တွေအရ အတွက်ကိုကြည့်ဖို့ဆိုရင် ကျွန်တော်တို့ ဒီလိုလုပ်နိုင်ပါတယ်။ ။

```
jok3r@lucy:~$ ls /bin /usr/bin | sort | uniq | wc -l
1894
jok3r@lucy:~$
```

grep—Print Lines Matching a Pattern

grep ဟာ powerful program တခုဖြစ်ပြီးတော့ file တွေထဲက text pattern တွေကိုရှာတဲ့အခါမှာ အသုံးပြုပါတယ်။ ။

grep pattern [file...]

grep က file ထဲမှာ pattern တခုကိုတွေ့တဲ့အခါ အဲ့ဒါတွေ (pattern) ပါတဲ့ line တွေအကုန်လုံးကို ထုတ်ပြပါတယ်။ ။ grep ကနေ match လုပ်ပေးနိုင်တဲ့ (တိုက်ပြီးရှာပေး) pattern တွေကရှုပ်ရှုပ်ထွေးထွေးတွေအထိ ရှိနေနိုင်ပေမယ့်အခုအတွက်တော့ကျွန်တော်တို့က ရိုးရိုး simple text match လုပ်တာလေးလောက်အပေါ်မှာပဲ အာရုံစိုက်ထားပါမယ်။ ။ ကျွန်တော်တို့ Chapter 19 ကျမှပဲ regular expression လို့ခေါ်တဲ့ အဆင့်မြင့် pattern တွေ အကြောင်းကိုပြည့်ပြည့်စုံစုံပြောမှာဖြစ်ပါတယ်။ ။

ကျွန်တော်တို့ရဲ့ list ထဲကရှိသမျှ file တွေထဲမှာ နာမည်မှာ zip ဆိုတဲ့စကားလုံးပါတဲ့ program တွေကိုရှာ ချင်တယ်ဆိုကြပါစို့။ ။ ဒီလိုရှာဖွေခြင်းကကျွန်တော်တို့ system ပေါ်ကဘယ် program တွေက file compression နဲ့ ပတ်သက်နေသလဲဆိုတာကိုသိမြင်စေပါတယ်။ ။ ကျွန်တော်တို့ကဒီလိုလုပ်ပါမယ်။ ။

```
jok3r@lucy:~$ ls /bin /usr/bin | sort | uniq | grep zip
bunzip2
bzip2
bzip2recover
funzip
gzip
```

```
gunzip
gzip
mzip
preunzip
prezip
prezip-bin
unzip
unzipsfx
zip
zipcloak
zipdetails
zipgrep
zipinfo
zipnote
zipsplit
jok3r@lucy:~$
```

ဒီနေရာမှာ grep အတွက်အသုံးဝင်တဲ့ option အချို့ရှိပါတယ် ။ -i ။ သူက grep အနေနဲ့ search လုပ်တဲ့ အခါမှာ case စာလုံးအကြီးအသေးမခွဲခြားပဲရှာပေးအောင်လုပ်စေပါတယ် ။ (သာမန်အားဖြင့် grep ကစာလုံးအကြီး အသေးခွဲခြားပြီးရှာပေးပါတယ်) ပြီးတော့ -v သူက grep ကို ရှာခိုင်းတဲ့ pattern နဲ့မတူတဲ့ line တွေကိုပဲ print ထုတ်ပြန်ပြောပေးပါတယ် ။

head/tail—Print First/Last Part of Files

တခါတလေမှာခင်ဗျားအနေနဲ့ command တခုဆီကနေ output တွေအကုန်လုံးကိုလိုချင်မှာမဟုတ်ပါဘူး ။ ခင်ဗျားအနေနဲ့ ပထမဆုံးသော line အနည်းငယ် သို့မဟုတ် နောက်ဆုံးသော line အနည်းငယ်ကိုပဲလိုချင်တာမျိုးရှိ ကောင်းရှိနိုင်ပါတယ် ။ head command က file ရဲ့ ပထမဦးဆုံးသော ၁၀ line ကိုဖော်ပြပေးပြီးတော့ tail command က နောက်ဆုံး ၁၀ line ကိုဖော်ပြပေးပါတယ် ။ default အရတော့ command ၂ ခုစလုံးက text ၁၀ လိုင်းစီကိုထုတ်ပြပေးပါတယ် ။ ဒါပေမယ့် ဒါကို -n option သုံးပြီးတော့ adjust လုပ်လို့ရပါတယ် ။

```
jok3r@lucy:~$ head -n 5 ls-output.txt
total 392624
-rwxr-xr-x 1 root root 59736 Sep  5 2019 [
-rwxr-xr-x 1 root root 96 Mar  8 2021 2to3-2.7
```

```

-rwxr-xr-x 1 root root    31248 May 19  2020 aa-enabled
-rwxr-xr-x 1 root root    35344 May 19  2020 aa-exec
jok3r@lucy:~$ tail -n 5 ls-output.txt
-rwxr-xr-x 1 root root    26952 Jan 31  2020 zjsdecode
-rwxr-xr-x 1 root root     2206 Apr  8 17:35 zless
-rwxr-xr-x 1 root root     1842 Apr  8 17:35 zmore
-rwxr-xr-x 1 root root     4577 Apr  8 17:35 znew
lrwxrwxrwx 1 root root         22 Jun  8  2021 zoom -> /opt/zoom/ZoomLauncher
jok3r@lucy:~$

```

ဒါကို pipeline တွေမှာလည်းအသုံးပြုလို့ရပါတယ် ။

```

jok3r@lucy:~$ ls /usr/bin | tail -n 5
zjsdecode
zless
zmore
znew
zoom
jok3r@lucy:~$

```

tail မှာ file တွေကို real time (အလုပ်လုပ်နေစဉ်) မှာကြည့်လို့ရတဲ့ option တခုရှိပါတယ် ။ ဒါက log file တွေရေးနေတုန်းသွားကြည့်ဖို့အတွက်အသုံးဝင်ပါတယ် ။ အောက်ကဥပမာထဲမှာကျွန်တော်တို့က /var/log ထဲက syslog file တွေကိုသွားကြည့်မှာဖြစ်ပါတယ် ။ ဒါကိုလုပ်ဖို့အတွက်တချို့ Linux distribution တွေမှာ Superuser privileges လိုအပ်ပါလိမ့်မယ် ။ ဘာလို့လဲဆိုတော့ /var/log/syslog file တွေမှာ security information လုံခြုံရေးဆိုင်ရာအချက်အလက်တွေပါဝင်ကောင်းပါဝင်နေနိုင်လို့ပါ ။

(စာအုပ်မူရင်းရေးသားသူက ဒီနေရာမှာ /var/log/messages ဆိုတဲ့ file ကိုသုံးပြီးလုပ်ပြထားပေမယ့် နောက်ပိုင်းထွက်လာတဲ့ Ubuntu version တွေမှာ အဲ့ဒါကိုမသုံးတော့တဲ့အတွက် လက်ရှိ log file တွေဖတ်တာကိုသုံးတဲ့ var/log/syslog နဲ့ပဲလုပ်ပြထားပါတယ် ။ /var/log/messages က non critical message အရေးမကြီးတဲ့ log တွေကိုသိမ်းပေးပြီးတော့ /var/log/syslog ကတော့ critical ရော non critical message ရဲ့ log တွေအကုန်လုံးကိုပါသိမ်းပေးထားပါတယ် ။)

```
jok3r@lucy:~$ sudo tail -f /var/log/syslog
```

```
[sudo] password for jok3r:
```

```
May  6 22:38:16 lucy systemd[2024]: tracker-store.service: Succeeded.
```

```
May  6 22:39:12 lucy gnome-shell[2183]: Window manager warning: Overwriting existing binding of  
keysym 31 with keysym 31 (keycode a).
```

```
May  6 22:39:12 lucy gnome-shell[2183]: Window manager warning: Overwriting existing binding of  
keysym 32 with keysym 32 (keycode b).
```

```
May  6 22:39:12 lucy gnome-shell[2183]: Window manager warning: Overwriting existing binding of  
keysym 33 with keysym 33 (keycode c).
```

```
May  6 22:39:12 lucy gnome-shell[2183]: Window manager warning: Overwriting existing binding of  
keysym 38 with keysym 38 (keycode 11).
```

```
May  6 22:39:12 lucy gnome-shell[2183]: Window manager warning: Overwriting existing binding of  
keysym 39 with keysym 39 (keycode 12).
```

```
May  6 22:39:12 lucy gnome-shell[2183]: Window manager warning: Overwriting existing binding of  
keysym 34 with keysym 34 (keycode d).
```

```
May  6 22:39:12 lucy gnome-shell[2183]: Window manager warning: Overwriting existing binding of  
keysym 35 with keysym 35 (keycode e).
```

```
May  6 22:39:12 lucy gnome-shell[2183]: Window manager warning: Overwriting existing binding of  
keysym 36 with keysym 36 (keycode f).
```

```
May  6 22:39:12 lucy gnome-shell[2183]: Window manager warning: Overwriting existing binding of  
keysym 37 with keysym 37 (keycode 10).
```

-f option ကိုသုံးလိုက်ခြင်းအားဖြင့် tail က file ကို ဆက်တိုက် monitor လုပ်နေမှာဖြစ်ပြီးတော့ line အသစ်တက်လာတာနဲ့သူကချက်ချင်းဖော်ပြပေးမှာဖြစ်ပါတယ် ။ ဒါက ခင်ဗျားအနေနဲ့ CTRL -C မနှိပ်မချင်းဆက်လုပ်နေမှာဖြစ်ပါတယ် ။

tee—Read from Stdin and Output to Stdout and Files

ကျွန်တော်တို့ရဲ့ plumbing analogy ပိုက်ဆက်ခြင်းနဲ့ပတ်သက်တဲ့တွေးခေါ်ဆင်ခြင်မှုတွေပြုလုပ်နေစဉ်မှာပဲ (pipeline command တွေနဲ့လုပ်နေလို့ ပိုက်ဆက်ခြင်း plumbing analogy လို့သုံးလိုက်တာပါ ။ ကြောင်မသွားပါနဲ့) Linux ကနေ ကျွန်တော်တို့ရဲ့ pipe တွေအပေါ်မှာ “T” fitting တွေ ဖန်တီးပေးတဲ့ tee ဆိုတဲ့ command တခုကို ထည့် သွင်းပေးထားပါတယ် ။ tee program က standard input ကိုဖတ်ပေးပြီးတော့ standard output ၊ ခုစလုံးဆီcopy တွေကူးပေးပါတယ် (data တွေကို pipeline တလျှောက်ဆက်လက်စီးဆင်းသွားခွင့်ပြုလိုက်ပြီးတော့) file တခု သို့မဟုတ် တခုထက်ပိုပြီးတော့ရှိနေတဲ့ဆီကိုပေါ့ ။ ဒါက pipeline ရဲ့ content တွေကို processing လုပ်နေတဲ့

intermediate stage မှာဖမ်းယူဖို့အတွက် အသုံးဝင်ပါတယ် ။ ဒီမှာကျွန်တော်တို့က အစောပိုင်းကဥပမာတစ်ခုကိုပြန် လုပ်ကြပါမယ် ။ အခုတကြိမ်မှာတော့ grep က pipeline ရဲ့ content တွေကို filter မလုပ်ခင်မှာ tee ကိုထည့်လိုက် ပြီးတော့ directory listing တစ်ခုကို ls.txt ဆိုတဲ့ file ထဲကိုသွားထည့်မှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ ls /usr/bin | tee ls.txt | grep zip
```

```
bunzip2
```

```
bzip2
```

```
bzip2recover
```

```
funzip
```

```
gpg-zip
```

```
gunzip
```

```
gzip
```

```
mzip
```

```
preunzip
```

```
prezip
```

```
prezip-bin
```

```
unzip
```

```
unzipsfx
```

```
zip
```

```
zipcloak
```

```
zipdetails
```

```
zipgrep
```

```
zipinfo
```

```
zipnote
```

```
zipsplit
```

```
jok3r@lucy:~$
```

Final Note

ခါတိုင်းလိုပဲ ၊ ကျွန်တော်တို့အခု chapter မှာလေ့လာခဲ့ပြီးသမျှ command တွေရဲ့ documentation page (man page) ကိုတချက်သွားကြည့်လိုက်ပါ ။ ကျွန်တော်တို့ဟာသူတို့တွေရဲ့အခြေခံအကျဆုံးအသုံးပြုပုံတွေကိုပဲ မြင် ရသေးတာဖြစ်ပြီးတော့ သူတို့တွေမှာစိတ်ဝင်စားစရာ option တွေအများအပြားရှိနေပါတယ် ။ ကျွန်တော်တို့ Linux အတွေ့အကြုံတွေများလာတာနဲ့အမျှကျွန်တော်တို့အနေနဲ့ command line ရဲ့ redirection feature ဟာspecialized

problem တွေကိုဖြေရှင်းရာမှာအလွန်အမင်းအသုံးဝင်တယ်ဆိုတာကိုသိမြင်လာပါလိမ့်မယ် ။ command အများစုက standard input နဲ့ output ကိုအသုံးပြုနေကြပြီးတော့ command-line program တွေအကုန်လုံးနီးပါးက သူတို့ရဲ့ informative message တွေကိုဖော်ပြဖို့အတွက် standard error ကိုအသုံးပြုကြပါတယ် ။

LINUX IS ABOUT IMAGINATION

ကျွန်တော့်ကို Windows နဲ့ Linux ကွာခြားချက်ကိုရှင်းပြခိုင်းတဲ့အချိန်တိုင်းမှာကျွန်တော်က ကစားစရာစဉ်းစားနည်းကိုအသုံးပြုပါတယ် ။

Windows ဟာ Game Boy ဂိမ်းစက် နဲ့တူပါတယ် ။ ခင်ဗျားက store ကိုသွားပြီးတော့ ဗူးထဲမှာအကုန်တောက်ပနေတဲ့ ဟာတချက်ကိုဝယ်လာတယ်ပေါ့ ။ ခင်ဗျားအိမ်ပြန်လာတယ် ။ စက်ဖွင့်ပြီးတော့အဲ့ဒါနဲ့ကစားတယ် ။ graphic လှလှကလေးတွေ ၊ ချစ်စရာအသံလေးတွေ ။ အချိန်တချက်ကြာလာတဲ့အခါကျတော့ ၊ ခင်ဗျားကစက်နဲ့အတူပါလာတဲ့ဂိမ်းကိုငြီးငွေ့လာပြီးတော့ store ကိုပြန်သွားပြီးနောက်ထပ် ဂိမ်းအသစ်တခုထပ်ဝယ်လိုက်တယ် ။ ဒီသံသရာကြီးကထပ်ခါထပ်ခါဖြစ်နေတော့တာပါပဲ ။ နောက်ဆုံးမှာတော့ခင်ဗျားက store ကိုပြန်သွားပြီးတော့ကောင်တာနောက်ကလူကိုမေးလိုက်ပါတော့တယ် ။ "ဟိုလိုဆော့တဲ့ဂိမ်းမရှိဘူးလားဗျ" ။ ခင်ဗျားကိုပြန်ပြောလိုက်မယ့်တခုထဲသောစကားကတော့ " အဲ့လိုဂိမ်းမျိုးမရှိပါဘူး ဘာလို့လဲဆိုတော့ အဲ့လိုဂိမ်းမျိုးအတွက်ဈေးကွက်တောင်းဆိုမှုမရှိလို့ပါ " လို့ဖြစ်ပါတယ် ။ အဲ့ဒီအခါမှာခင်ဗျားက "ဒါပေမယ့်ကျွန်တော်က (ဂိမ်းထဲက) ဒါလေးကိုပဲပြောင်းလိုက်ချင်တာလေ" လို့ပြောပါမယ် ။ အဲ့ဒီအခါမှာ ကောင်တာနောက်ကလူက မင်းပြောင်းလို့မရဘူးလေလို့ပြောပါလိမ့်မယ် ။ ဂိမ်းတွေက cartridge တွေထဲမှာ (အသေ) seal ပိတ်ထားတာပါ ။ ခင်ဗျားရဲ့ကစားစရာဟာဂိမ်းတွေအတွက်ကန့်သတ်ချက်ရှိနေတယ်ဆိုတာကိုခင်ဗျားရှာဖွေတွေ့ရှိသွားပြီးတော့အခြားသူတွေကတော့ခင်ဗျားအနေနဲ့နောက်ထပ်နောက်ထပ် (ဂိမ်းတွေ) ဝယ်ယူဖို့လိုတယ်လို့ဆုံးဖြတ်ကြမှာဖြစ်ပါတယ် ။

အခြားတဖက်မှာတော့ ၊ Linux ဟာ ကမ္ဘာ့အကြီးဆုံး Erector Set (lego တို့လိုအရုပ်ဆက်ကစားစရာ) တခုဖြစ်ပါတယ် ။ ခင်ဗျား (ဗူးကို) ဖွင့်ဖောက်လိုက်တာနဲ့အစိတ်အပိုင်းတွေအများကြီးပါတဲ့ ကြီးမားတဲ့ collection တခု --- steel struts အပေါက်ပါတဲ့သံပြားလေးတွေအများကြီး ၊ စကူတွေ ၊ နပ်တွေ ၊ ဂီယာတွေ ၊ ရစ်ဘီးတွေ ၊ မော်တာတွေ နဲ့ ဘာတွေဆောက်ရမယ် (ဘာတွေဆောက်လို့ရတယ်) ဆိုတာအနည်းငယ်ပါတဲ့ (ပြောပြထားတဲ့) အကြံပြုချက်တခုပါဝင်ပါလိမ့်မယ် ။ ဒါကြောင့်ခင်ဗျားစတင်ပြီးတော့ကစားပါတယ် ။ ခင်ဗျားက အကြံပြုချက်ထဲကတခုကိုဆောက်ကြည့်ပြီးတော့နောက်တခုဆက်ဆောက်ပါတယ် ။ ခဏကြာတဲ့အခါမှာခင်ဗျားမှာဘာဆောက်မယ်ဆိုတဲ့ခင်ဗျားကိုယ်ပိုင်စိတ်ကူးရလာပါတယ် ။ ခင်ဗျားအနေနဲ့ store ကိုပြန်သွားစရာမလိုတော့ပါဘူး ။ ဘာလို့လဲဆိုတော့ခင်ဗျားမှာအကုန်ရှိနေပြီဖြစ်လို့ပါပဲ ။ Erector Set ကခင်ဗျားရဲ့စိတ်ကူးကိုပုံဖော်လုပ်ဆောင်ပေးပါတယ် ။ ခင်ဗျားဖြစ်ချင်တာကိုသူကလုပ်ပေးပါတယ် ။

ခင်ဗျားရဲ့ကစားစရာရွေးချယ်မှုဟာ ဟုတ်ပါတယ် ခင်ဗျားရဲ့ကိုယ်ရေးကိုယ်တာကိစ္စပါ ။ အဲ့ဒီတော့ ဘယ်ကစားစရာကိုခင်ဗျားကပိုပြီးစိတ်ကျေနပ်မှုရှိပါသလဲ ?

SEEING THE WORLD AS THE SHELL SEES IT

အခု chapter မှာကျွန်တော်တို့က ခင်ဗျား ENTER key ကိုနှိပ်လိုက်တဲ့အချိန်မှာဖြစ်ပေါ်လာမယ့် ဆန်းကြယ်မှု Magic တချို့ကိုလေ့လာ ကြည့်ကြမှာဖြစ်ပါတယ်။ ကျွန်တော်တို့က shell ရဲ့ စိတ်ဝင်စားစရာကောင်း ပြီးတော့ရှုပ်ထွေးလှတဲ့ feature ကိုစမ်းစစ်လေ့လာနေစဉ်မှာပဲအဲ့ဒါကို command အသစ် တခုထဲနဲ့လုပ်သွားမှာဖြစ်ပါတယ်။

- **echo** — text တွေကို တ line စာ ဖော်ပြပေးမယ်။

Expansion

ခင်ဗျားအနေနဲ့ command line တခုရှိပြီး Enter key နှိပ်လိုက်တိုင်းမှာ bash က ခင်ဗျားရဲ့ command ကိုအလုပ်မလုပ်ပေးခင် text ပေါ်မှာ process အများအပြားကိုလုပ်ပေးရပါသေးတယ်။ ကျွန်တော်တို့အနေနဲ့ simple character တခုက အစဉ်လိုက်ဘယ်လိုစီပေးသလဲဆိုတဲ့ကိစ္စတွေတော်တော်များများကိုတွေ့ခဲ့ပြီးဖြစ်ပါတယ်။ ဥပမာ * ဆိုရင် shell အတွက်အဓိပ္ပါယ်တွေအများကြီးရှိနေပါတယ်။ အဲ့ဒါကိုဖြစ်အောင်လုပ်ပေးတဲ့ process ကို expansion လို့ခေါ်ပါတယ်။ expansion နဲ့ဆိုရင် ခင်ဗျားအနေနဲ့တခုခုရှိထည့်လိုက်ပြီးတော့ shell ကသူ့အပေါ်မှာအလုပ်မလုပ်ခင်မှာ အဲ့ဒီအရာကအခြားအရာတခုခုအဖြစ် expanded ဖြစ်သွားစေပါတယ်။ ကျွန်တော်ဆိုလိုတာကိုလက်တွေ့ပြရမယ်ဆိုရင်တော့ echo command ကိုတချက်လောက်ကြည့်ကြစို့။ echo ဆိုတာ shell builtin တခုဖြစ်ပြီးတော့သူကအလွန်ရိုးရှင်းတဲ့ လုပ်ငန်းတာဝန်တခုကိုပဲလုပ်ဆောင်ပေးပါတယ်။ သူကသူ့ရဲ့ text argument တွေကို standard output ပေါ်ကိုထုတ်ပြပေးပါတယ်။

```
jok3r@lucy:~$ echo this is a test
this is a test
jok3r@lucy:~$
```

ဒါကတော်တော်လေးတဲ့တိုးဆန်ပါတယ်။ echo ဆီကို pass လုပ်လိုက်တဲ့ argument တွေမှန်သမျှကို ဖော်ပြပေးပါတယ်။ နောက်ထပ်ဥပမာတခုထပ်လုပ်ကြည့်ကြစို့။

```
jok3r@lucy:~$ echo *
```

```
default.ovpn Desktop Documents Downloads Dropbox dwhelper hacking_tutorial_link  
javasharedresources master.zip MEGAsync mm-kb-master Music Pictures pt Public Recordings snap  
Templates Videos
```

```
jok3r@lucy:~$
```

ဒါဆို အခုဘာဖြစ်သွားတာလဲ ? echo ကဘာဖြစ်လို့ * ပုံကို print မထုတ်ပေးတာလဲ ? ခင်ဗျားအနေနဲ့ wildcard တွေနဲ့လုပ်ခဲ့တုန်းကဟာကိုပြန်မှတ်မိမယ်ဆိုရင် * character ဆိုတာက "file name ထဲကမည်သည့် character နှင့်မဆိုတိုက်ဆိုင်စစ်ဆေးပေးပါ " လို့ဖြစ်ပေမယ့် ကျွန်တော်တို့အနေနဲ့ မူရင်းဆွေးနွေးခဲ့စဉ်က shell က အဲ့ဒါ (* character ကို) ဘယ်လိုအလုပ်လုပ်ပေးသလဲဆိုတာကိုမမြင်ခဲ့ရပါဘူး ။ အရိုးရှင်းဆုံးအဖြေကတော့ shell က * character ကိုအခြားအရာတခုအဖြစ် echo command က execute မလုပ်ခင်မှာ expand လုပ်လိုက်ပါတယ် ။ (အခုကိစ္စမှာဆိုရင်တော့ current working directory ထဲက file တွေရဲ့နာမည်တွေကိုပေါ့) ENTER key ကိုနှိပ်လိုက်တဲ့အခါမှာ shell က command ဆက်ပြီးအလုပ်မလုပ်ခင်မှာ command-line ပေါ်က သတ်မှတ်ချက်ပြည့်မှီတဲ့ character တွေကို အလိုအလျောက် expand လုပ်လိုက်ပါတယ် ။ ဒါကြောင့် echo command က * ကိုမမြင်လိုက်ပဲနဲ့သူ့ရဲ့ expand လုပ်ထားတဲ့ result ကိုပဲမြင်ပါတယ် ။ တခုတော့သိထားပါ ။ echo ကကျွန်တော်တို့မျှော်လင့်ထားတဲ့အတိုင်း (ထင်တဲ့အတိုင်း) အလုပ်လုပ်ပေးပါတယ် ။

Pathname Expansion

wildcard တွေအတွက်လုပ်ပေးတဲ့ mechanism ကို pathname expansion လို့ခေါ်ပါတယ် ။ ကျွန်တော်တို့ရဲ့အစောပိုင်း chapter တွေမှာ ကျွန်တော်တို့အသုံးချခဲ့တဲ့နည်းစနစ်တွေထဲကအချို့ကိုပြန်လေ့လာကြည့်မယ်ဆိုရင်ကျွန်တော်တို့အနေနဲ့အဲ့ဒါတွေဟာတကယ့် Expansion တွေဆိုတာကိုမြင်လာပါလိမ့်မယ် ။ home directory ကိုအောက်ကလိုပုံစံမျိုးဖြစ်အောင်လုပ်ပေးပါ ။ (directory ဆောက်တဲ့အပေါ်မူတည်ပြီးတယောက်နဲ့တယောက်တူမှာမဟုတ်ပါဘူး ။ Os စသွင်းကထဲကပါတဲ့ default directory တွေပါရင်ရပါပြီ ။ အောက်မှာပြထားတာနဲ့တထပ်ထဲမတူလို့မှားနေတယ်လို့ကိုယ့်ကိုကိုယ်ထင်မှာစိုးလို့ပါ ။)

```
jok3r@lucy:~$ ls
```

```
default.ovpn Downloads hacking_tutorial_link MEGAsync Pictures Recordings Videos  
Desktop Dropbox javasharedresources mm-kb-master pt snap  
Documents dwhelper master.zip Music Public Templates
```

```
jok3r@lucy:~$
```

ကျွန်တော်တို့အနေနဲ့အောက်ပါ expansion တွေကိုဆက်လုပ်သွားလို့ရပါတယ် ။

```
jok3r@lucy:~$ echo D*
```

```
Desktop Documents Downloads Dropbox
```

```
jok3r@lucy:~$
```

ပြီးတော့

```
jok3r@lucy:~$ echo *s
```

```
Documents Downloads javasharedresources Pictures Recordings Templates Videos
```

```
jok3r@lucy:~$
```

တင်မကပဲ

```
jok3r@lucy:~$ echo [[:upper:]]*
```

```
Desktop Documents Downloads Dropbox MEGAsync Music Pictures Public Recordings Templates Videos
```

```
jok3r@lucy:~$
```

ပြီးတော့ကျွန်တော်တို့ရဲ့ home directory ကိုပါကျော်လွန်ပြီးကြည့်လို့ရပါသေးတယ် ။

```
jok3r@lucy:/$ echo /usr/*/share
```

```
/usr/local/share
```

PATHNAME EXPANSION OF HIDDEN FILES

ကျွန်တော်တို့သိတဲ့အတိုင်း period character (.) တခုနဲ့စတဲ့ filename တွေ (ရှိတဲ့ file တွေ directory တွေ) ဟာ hidden (ဖွက်ထားတဲ့ file တွေ directory တွေ) ဖြစ်ပါတယ် ။ path name expansion က အဲ့ဒီအပြုအမူ ဆောင်ရွက်ချက်ကိုလေးစားလိုက်နာပါတယ် ။ ဒီလိုမျိုး expansion

echo *

က hidden file တွေကိုမဖော်ပြပေးနိုင်ပါဘူး ။

သူကအစပိုင်းတွေတွေ့ချင်းမှာတော့ ကျွန်တော်တို့အနေနဲ့ expansion တခုထဲမှာ pattern ရဲ့အစမှာ period (.) တခုအရင်ထည့်ပြီးတော့ hidden file တွေပါ (ဖော်ပြအောင်လို့)ပါအောင်လုပ်လို့ရမယ်လို့ထင်စရာ အကြောင်းရှိပါတယ် ။ ဒီလိုမျိုးပေါ့

echo .*

အဲ့ဒါကအလုပ်ဖြစ်လုနီးပါးပါပဲ ။ ကျွန်တော်တို့အနေနဲ့ result တွေကိုအနီးကပ်သေချာစစ်ဆေးကြည့်မယ်ဆိုရင် result တွေထဲမှာ . နဲ့ .. ဆိုတဲ့နာမည်တွေတွေ့ရမှာဖြစ်ပါတယ် ။ အဲ့ဒီနာမည်တွေက current working directory နဲ့ သူ့ရဲ့ parent directory ကိုညွှန်ပြနေတာဖြစ်တဲ့အတွက် ဒီ pattern ကိုအသုံးပြုခြင်းဟာ အဖြေအမှားတွေထွက်လာဖို့များပါတယ် ။ ကျွန်တော်တို့ (မြင်ချင်သလို) မြင်နိုင်ပါတယ် ။ တကယ်လို့ကျွန်တော်တို့ကဒီ command နဲ့လုပ်ကြည့်မယ်ဆိုရင်ပေါ့ ။

ls -d .* | less

အခုအခြေအနေမှာ pathname expansion ကိုမှန်မှန်ကန်ကန်လုပ်ဆောင်စေဖို့အတွက်ဆိုရင်ကျွန်တော်တို့အနေနဲ့ ပိုပြီးတိကျတဲ့ pattern တခုကိုထည့်ပေးရမှာဖြစ်ပါတယ် ။

ls -d .[!.]? *

ဒီ pattern က period (.) တခုနဲ့စထားတဲ့ filename တိုင်းကို expand လုပ်သွားမှာဖြစ်ပါတယ် ။ ဒုတိယမြောက် period (.) မပါဝင်ပါဘူး ။ အနည်းဆုံး နောက်တိုး character တခုပါဝင်ပြီးတော့သူ့နောက်မှာ အခြား character တွေထပ်ပါချင်လည်းပါလာပါလိမ့်မယ် ။

Tilde Expansion

ခင်ဗျားအနေနဲ့ကျွန်တော်တို့ရဲ့ cd command ကိုမိတ်ဆက်ပေးစဉ်တုန်းကဟာတွေကိုပြန်မှတ်မိနေမယ်ဆိုရင် tilde character (~) မှာ special meaning တခုရှိနေပါတယ် ။ စကားလုံးတလုံးရဲ့အစမှာသုံးမယ်ဆိုရင် သူက နာမည်ပေးပြီးသား user ရဲ့ home directory အမည်ထဲကို ဒါမှမဟုတ် user တဦးမှမရှိခဲ့ရင် current user ရဲ့ home directory ထဲကို expand လုပ်ပေးသွားမှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ echo ~  
/home/jok3r  
jok3r@lucy:~$
```

တကယ်လို့ user foo အတွက် account တခုရှိနေခဲ့မယ်ဆိုရင်

```
jok3r@lucy:~$ echo ~foo
/home/foo
jok3r@lucy:~$
```

(တကယ်လို့ user foo အတွက် account တခု မရှိဘူးဆိုရင်တော့)

```
jok3r@lucy:~$ echo ~foo
~ foo
jok3r@lucy:~$
```

Arithmetic Expansion

shell က ဂဏန်းသင်္ချာဆိုင်ရာများကိုလည်း expansion အနေနဲ့လုပ်ဆောင်ခွင့်ပြုထားပါတယ် ။ ဒါက ကျွန်တော်တို့ကို shell prompt ကို calculator အနေနဲ့သုံးခွင့်ရစေပါတယ် ။

```
jok3r@lucy:~$ echo $((2+2))
4
jok3r@lucy:~$
```

Arithmetic expansion က အောက်ပါပုံစံကိုအသုံးပြုပါတယ် ။

\$ ((expression))

expression နေရာမှာ တန်ဖိုးတွေ (1,2,3 စသည်ဖြင့်) နဲ့ arithmetic operator (+,-,*,/,%, စသည်ဖြင့်) ပါဝင်တဲ့ arithmetic expression ဖြစ်ရမှာဖြစ်ပါတယ် ။

Arithmetic expansion က integer (ကိန်းပြည့် ၊ ဒဿမကိန်းမဖြစ်ရပါ) တခုထဲကိုပဲ support လုပ်ပေးပေမယ့်လုပ်ဆောင်ချက်အတော်များများကိုတော့လုပ်ပေးနိုင်စွမ်းရှိပါတယ် ။ Table 7-1 မှာသူ support ပေးတဲ့ operator တွေတချို့ကို list လုပ်ပြထားပါတယ် ။

Table 7-1: Arithmetic Operators

Operator	Description
+	ပေါင်းတာ
-	နှုတ်တာ
*	မြှောက်တာ
/	စားတာ (တခုမှတ်ထားရမှာက expansion ကကိန်းပြည့်ကိုပဲ support လုပ်တဲ့အတွက် အဖြေကလည်းကိန်းပြည့်နဲ့ပဲထွက်မှာပါ ။)
%	စားကြွင်းကိန်း
**	အကွေ့ရာထပ်ကိန်း ၊ ထပ်ညွှန်းကိန်း

arithmetic expression တွေမှာ space (ကွက်လပ်) တွေမပါရပါဘူး ။ ပြီးတော့ expression တွေဟာ nested (တခုပေါ်တခုထပ်ခွေ) ဖြစ်ချင်ဖြစ်နေနိုင်ပါတယ် ။ ဥပမာ - 5^2 ကို 3 နဲ့မြှောက်တာမျိုး ။

```
jok3r@lucy:~$ echo $(((5**2))*3))
75
jok3r@lucy:~$
```

တခုထက်ပိုတဲ့ subexpression တွေကို group လုပ်ဖို့အတွက် Single parenthese တွေကိုအသုံးပြုပါတယ် ။ ဒီနည်းစနစ်နဲ့ကျွန်တော်တို့ကအပေါ်က ဥပမာနဲ့အဖြေတူတူထွက်အောင် expansion ၂ ခုနေရာမှာ တခုထဲကို အသုံးပြုပြီးတော့ ပြန်ရေးမှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ echo $((((5**2))*3))
75
jok3r@lucy:~$
```

ဒီမှာ စားလဒ် နဲ့ စားကြွင်း operator တွေကိုအသုံးပြုထားတဲ့ ဥပမာပါ ။ ကိန်းပြည့်ကိုစားတဲ့အခါဘယ်လို သက်ရောက်မှုရှိသလဲဆိုတာကိုသတိထားကြည့်လိုက်ပါ ။

```
jok3r@lucy:~$ echo five divided by two equals  $\$(5/2)$ 
five divided by two equals 2
jok3r@lucy:~$ echo with  $\$(5\%2)$  left over.
with 1 left over.
jok3r@lucy:~$
```

Arithmetic expansion တွေအကြောင်းကို Chapter 34 ကျရင်ဒီထက်ပိုပြီးအသေးစိတ်လေ့လာသွားမှာဖြစ်ပါတယ်။ ။

Brace Expansion

အထူးဆန်းဆုံးဖြစ်ကောင်းဖြစ်နိုင်တဲ့ expansion ကို brace expansion လို့ခေါ်ပါတယ်။ ။ အဲ့ဒါနဲ့ဆိုရင် ခင်ဗျားအနေနဲ့ တွန့်ကွင်းတွေပါတဲ့ pattern တခုထဲက text string တွေအများအပြားကိုဖန်တီးလို့ရပါတယ်။ ။ ဥပမာကဒီမှာပါ။ ။

```
jok3r@lucy:~$ echo Front-{A,B,C}-Back
Front-A-Back Front-B-Back Front-C-Back
jok3r@lucy:~$
```

pattern တွေကို brace expansion လုပ်တဲ့အခါမှာ သူ့မတိုင်ခင်အရှေ့မှာ(ကိုယ်လိုချင်တာရေးထည့်လို့ရတဲ့) preamble လို့ခေါ်တဲ့ leading portion တခုနဲ့သူ့အဆုံးမှာ(ကိုယ်လိုချင်တာရေးထည့်လို့ရတဲ့) postscript လို့ခေါ်တဲ့ trailing portion တခု ပါဝင်ကောင်းပါဝင်နိုင်ပါတယ်။ ။ brace expression ရဲ့အထဲမှာတော့ comma (,) နဲ့ခြားထားတဲ့ String (စာသား) တွေ သို့မဟုတ် integer (ကိန်းပြည့်) အတွဲလိုက်တခု သို့မဟုတ် character တလုံးစီ ပါဝင်ကောင်းပါဝင်နိုင်ပါတယ်။ ။ pattern ထဲမှာ whitespace တွေမပါရပါဘူး။ ။ (space ပါသွားခဲ့ရင်အလုပ်မလုပ်ပေးပဲကိုယ်ရိုက်ထားတာကိုပြန်ထုတ်ပြနေပါလိမ့်မယ်။) ဒီမှာကိန်းပြည့်အစဉ်တန်း range of integer ကိုအသုံးပြုပြီး ဥပမာတခုလုပ်ပြထားပါတယ်။ ။

```
jok3r@lucy:~$ echo Number_{1..5}
Number_1 Number_2 Number_3 Number_4 Number_5
jok3r@lucy:~$
```

ဒီမှာကျွန်တော်တို့ စာလုံးတွေကိုပြောင်းပြန် စီပေးထားပါတယ်။ ။

```
jok3r@lucy:~$ echo {Z..A}
ZYXWVUTSRQPONMLKJIHGFEDCBA
jok3r@lucy:~$
```

Brace expansion တွေဟာ nested အတွင်းမှာခွေနေတာမျိုးလည်းရှိနိုင်ပါတယ် ။

```
jok3r@lucy:~$ echo a{A{1,2},B{3,4}}b
aA1b aA2b aB3b aB4b
jok3r@lucy:~$
```

ဒါဆိုအဲ့ဒါကဘာအတွက်ကောင်းတာလဲ ? application အများစုမှာကတော့ file တွေ သို့မဟုတ် directroy တွေရဲ့ list ကိုဖန်တီးရာမှာသုံးကြပါတယ် ။ ဥပမာ တကယ်လို့ ကျွန်တော်တို့က ဓာတ်ပုံဆရာတွေဖြစ်ပြီးတော့ ကျွန်တော်တို့မှာ နှစ်တွေ လတွေ အလိုက်စုစည်းချင်တဲ့ စုဆောင်းထားတဲ့ ဓာတ်ပုံတွေအများကြီးရှိတယ်ဆိုရင် ကျွန်တော်တို့အနေနဲ့ပထမဦးဆုံးလုပ်မိမှာကတော့ နှစ်လ တွေကိုဂဏန်းနဲ့နာမည်ပေးထားတဲ့ directroy အတွဲလိုက် တခုပဲဖြစ်ပါတယ် ။ ဒီလိုနည်းလမ်းနဲ့ဆိုရင် directroy တွေကို အချိန်နဲ့အစီအစဉ်အတိုင်း စဉ်ပေးသွားမှာဖြစ်ပါတယ် ။ ကျွန်တော်တို့အနေနဲ့ directroy တွေရဲ့ list ကို (လက်နဲ့)ရိုက်ထည့်ပေးလို့ရပေမယ့်အဲ့ဒါကအလုပ်များလွန်းပြီး တော့အမှားဖြစ်ဖို့လည်းများပါတယ် ။ အဲ့ဒီလိုလုပ်မယ့်အစား ကျွန်တော်တို့အနေနဲ့ဒီလိုလုပ်လို့ရပါတယ် ။

```
jok3r@lucy:~$ mkdir Pics
jok3r@lucy:~$ cd Pics
jok3r@lucy:~/Pics$ mkdir {2009..2011}-0{1..9} {2009..2011}-{10..12}
jok3r@lucy:~/Pics$ ls
2009-01 2009-05 2009-09 2010-01 2010-05 2010-09 2011-01 2011-05 2011-09
2009-02 2009-06 2009-10 2010-02 2010-06 2010-10 2011-02 2011-06 2011-10
2009-03 2009-07 2009-11 2010-03 2010-07 2010-11 2011-03 2011-07 2011-11
2009-04 2009-08 2009-12 2010-04 2010-08 2010-12 2011-04 2011-08 2011-12
jok3r@lucy:~/Pics$
```

အရမ်းမိုက်တယ်မဟုတ်လား !

Parameter Expansion

ကျွန်တော်တို့အနေနဲ့ အခု Chapter မှာ parameter expansion ကိုအကြမ်းမျဉ်းလောက်ပဲကိုင်တွယ်သွားမှာဖြစ်ပြီးတော့ကျွန်တော်တို့နောက်ကျမှအကျယ်တဝင့်လေ့လာကြပါမယ် ။ သူက shell script ပေါ်မှာထက် command-line ပေါ်မှာပိုပြီးအသုံးဝင်တဲ့ feature တခုဖြစ်ပါတယ် ။ data အပိုင်းအစလေးတွေကိုသိုလှောင်ထားပေးပြီးတော့ data အပိုင်းအစတခုချင်းစီကိုနာမည်ပေးဖို့အတွက် သူ့ရဲ့လုပ်နိုင်စွမ်းဟာ system ရဲ့စွမ်းဆောင်ရည်အပေါ်မူတည်ပက်သက်နေပါတယ် ။ အပိုင်းအစတွေအများစု ၊ ပိုပြီးမှန်ကန်အောင်ခေါ်ရမယ်ဆိုရင် variable တွေဟာ ခင်ဗျားကြည့်ရှုစစ်ဆေးဖို့အတွက် အဆင်သင့်ဖြစ်နေကြပါတယ် ။ ဥပမာ - USER ဆိုတဲ့ variable name ထဲမှာ ခင်ဗျားရဲ့ username ရှိနေပါတယ် ။ parameter expansion ကို invoke လုပ်ပြီးတော့ USER ထဲက content တွေကိုဖော်ထုတ်ဖို့အတွက်ဆိုရင် ခင်ဗျားအနေနဲ့ဒီလိုလုပ်ရမှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ echo $USER
jok3r
jok3r@lucy:~$
```

ရှိသမျှ variable တွေအကုန်ရဲ့ list ကိုကြည့်ဖို့အတွက်ဆိုရင် ၊ ဒီလိုလုပ်ကြည့်ပါ ။

```
jok3r@lucy:~$ printenv | less
```

ခင်ဗျားအနေနဲ့သတိထားမိချင်ရင်ထားမိမယ်ထင်ပါတယ် ။ အခြား expansion type တွေနဲ့ဆိုရင် တကယ်လို့ခင်ဗျားက pattern ကိုမှားပြီးရိုက်ထည့်မိရင် expansion ကအလုပ်မလုပ်တော့ပဲနဲ့ echo command က ရိုးရိုးလေး ခင်ဗျားမှားရိုက်ထည့်ထားတာကိုပဲပြန်ပြပေးနေပါလိမ့်မယ် ။ parameter expansion နဲ့ဆိုရင် တကယ်လို့ ခင်ဗျားက variable ရဲ့နာမည်ကိုစာလုံးပေါင်းမှားပြီးရိုက်ထည့်လိုက်ရင် expansion ကဆက်ပြီးအလုပ်လုပ်ပေးနေမှာ ဖြစ်ပေမယ့် result ကတော့ empty string (ဘာမှမပေါ်တဲဟာ) ဖြစ်နေပါလိမ့်မယ် ။

```
jok3r@lucy:~$ echo $SUER
jok3r@lucy:~$
```

Command Substitution

Command substitution ကကျွန်တော်တို့ကို command တခုရဲ့ output ကို expansion အနေနဲ့အသုံးပြုခွင့်ပေးပါတယ် ။

```
jok3r@lucy:~$ echo $(ls)
```

```
default.ovpn Desktop Documents Downloads Dropbox dwhepler hacking_tutorial_link javasharedresources  
master.zip MEGAsync mm-kb-master Music Pictures pt Public Recordings snap Templates Videos
```

```
jok3r@lucy:~$
```

ကျွန်တော်အကြိုက်ဆုံးတွေထဲကတခုကတော့ဒီလိုဟာမျိုးပါ။ ။

```
jok3r@lucy:~$ ls -l $(which cp)
```

```
-rwxr-xr-x 1 root root 153976 Sep 5 2019 /usr/bin/cp
```

```
jok3r@lucy:~$
```

ဒီနေရာမှာကျွန်တော်တို့က which cp ရဲ့ result ကို argument အနေနဲ့ ls command ဆီကိုပို့ပေးပါတယ် ။ ဒါကြောင့်မို့ cp program ရဲ့ listing ကိုသူ့ရဲ့ pathname အပြည့်အစုံသိစရာမလိုပဲရရှိနိုင်ပါတယ် ။ ကျွန်တော်တို့ဟာဒီလို ရိုးရှင်းတဲ့ command တွေမှာပဲကန့်သတ်(ရပ်တန့်)နေမှာမဟုတ်ပါဘူး ။ pipeline တွေအကုန်လုံးကိုလည်းအသုံးပြုလို့ရပါတယ် ။ (စာမျက်နှာအခက်အခဲကြောင့် result တချို့ကိုသာဖော်ပြထားပါတယ် ။)

```
jok3r@lucy:~$ file $(ls /usr/bin/* | grep zip)
```

```
/usr/bin/bunzip2:      ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter  
/lib64/ld-linux-x86-64.so.2, BuildID[sha1]=00c09d800d549d58e3b7c4f6170446cc69bf14a5, for GNU/Linux 3.2.0,  
stripped
```

```
/usr/bin/bzip2:        ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter  
/lib64/ld-linux-x86-64.so.2, BuildID[sha1]=00c09d800d549d58e3b7c4f6170446cc69bf14a5, for GNU/Linux 3.2.0,  
stripped
```

```
/usr/bin/bzip2recover: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter  
/lib64/ld-linux-x86-64.so.2, BuildID[sha1]=aa9666f913c3a67eab9d62585a9b37e46d0c7cb9, for GNU/Linux 3.2.0,  
stripped
```

```
/usr/bin/funzip:       ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter  
/lib64/ld-linux-x86-64.so.2, BuildID[sha1]=16c10d3e99879c6257b23fba3c49032d8ac2e5b4, for GNU/Linux 3.2.0,  
stripped
```

```
/usr/bin/gpg-zip:      POSIX shell script, ASCII text executable
```

```
/usr/bin/gunzip:       POSIX shell script, ASCII text executable
```

/usr/bin/gzip: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=cbba8a5cb33da77dc39f4cf1c00beb8a680abbd9, for GNU/Linux 3.2.0, stripped

ဒီဥပမာထဲမှာ pipeline ရဲ့ result က file command ရဲ့ argument list ဖြစ်သွားပါတယ် ။

ဒီနေရာမှာ bash မှာလည်း support လုပ်ပေးတဲ့ shell program အဟောင်းထဲက command ကိုအစားထိုးဖို့အတွက် အခြားရွေးချယ်စရာ syntax တခုလည်းရှိပါတယ် ။ သူက dollar sign (\$) နဲ့ကွင်းစကွင်းပိတ်တွေအစား back quotes (`) တွေကိုအသုံးပြုပါတယ် ။

```
jok3r@lucy:~$ ls -l `which cp`  
-rwxr-xr-x 1 root root 153976 Sep  5 2019 /usr/bin/cp  
jok3r@lucy:~$
```

Quoting

အခုဆိုရင်ကျွန်တော်တို့ဟာ shell က နည်းလမ်းပေါင်းဘယ်လောက်များများနဲ့အလုပ်လုပ်ပေးနိုင်သလဲဆိုတာကိုတွေ့ခဲ့ပါပြီ ။ အခု သူ့ကိုကျွန်တော်တို့ဘယ်လို control လုပ်ရမယ်ဆိုတာကိုလေ့လာဖို့အချိန်ကျပါပြီ ။ ဥပမာ - ဒါကိုလုပ်ကြည့်ပါ ။

```
jok3r@lucy:~$ echo this is a test  
this is a test  
jok3r@lucy:~$
```

ဒါမှမဟုတ် ဒီလို

```
jok3r@lucy:~$ echo The total is $100.00  
The total is 00.00  
jok3r@lucy:~$
```

ပထမဥပမာထဲမှာ shell က word splitting နဲ့ echo command ရဲ့ argument list ထဲက whitespace တွေကိုဖယ်ရှားပစ်လိုက်ပါတယ် ။ ဒုတိယဥပမာထဲမှာ parameter expansion က \$1 အတွက် empty string နဲ့

အစားထိုးလိုက်ပါတယ် ။ ဘာလို့လဲဆိုတော့ သူက undefine variable (သုံးဖို့အတွက်မကြေငြာပေးရသေးတဲ့ variable) တွေဖြစ်နေလို့ပါပဲ ။ shell မှာ မလိုလားအပ်တဲ့ expansion တွေကိုရွေးချယ်ပြီး ဖိနှိပ်ကန့်သတ်ထားဖို့ အတွက် quoting လို့ခေါ်တဲ့ mechanism တခုထည့်ထားပေးပါတယ် ။

Double Quotes

ကျွန်တော်တို့ကြည့်ကြမယ့်ပထမဆုံးသော quoting ကတော့ double quote ဖြစ်ပါတယ် ။ တကယ်လို့ ခင်ဗျားက text တွေကို double quoteအတွင်းမှာထည့်လိုက်မယ်ဆိုရင် shell ကအသုံးပြုနေတဲ့ special character တွေအကုန်လုံးဟာသူတို့ရဲ့ special meaning တွေပျောက်ကုန်ပြီးတော့ရိုးရိုး character တွေအနေနဲ့ပဲ ဆက်ဆံခံရပါတော့တယ် ။ \$ (dollar sign), \ (backslash) နဲ့ ` (back tick) တို့ကိုတော့ ချင်းချက်အနေနဲ့ထားခွင့် ပေးပါတယ် ။ ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ word splitting , pathname expansion, tilde expansion နဲ့ brace expansion တို့က ဖိနှိပ်ကန့်သတ်ခံရပေမယ့် parameter expansion, arithmetic expansion နဲ့ command substitution တို့ကတော့အလုပ်ဆက်လုပ်နေတယ်ဆိုတဲ့သဘောပါပဲ ။ double quote တွေကိုသုံးခြင်းအားဖြင့် ကျွန်တော်တို့အနေနဲ့ embedded space တွေပါဝင်တဲ့ filename တွေကိုစီမံခန့်ခွဲလို့ရသွားပါတယ် ။ ဆိုကြပါစို့ ကျွန်တော်တို့က two words.txt ဆိုတဲ့ file တခုရဲ့ ကံမကောင်းလှစွာသောသားကောင်တွေပေါ့ ။ တကယ်လို့ ကျွန်တော်တို့က ဒါကို command-line ပေါ်မှာသုံးလိုက်ခဲ့ရင် word splitting ကနေကျွန်တော်တို့လိုချင်တဲ့စကားလုံး တဆက်ထဲ single argument အနေနဲ့မဟုတ်ပဲ မတူကွဲပြားတဲ့ argument ၂ ခုအနေနဲ့ဆက်ဆံခြင်းကိုဖြစ်ပေါ်စေ မှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ ls -l two words.txt
ls: cannot access 'two': No such file or directory
ls: cannot access 'words.txt': No such file or directory
jok3r@lucy:~$
```

double quote ကိုသုံးခြင်းအားဖြင့်ကျွန်တော်တို့အနေနဲ့ word splitting ကိုတားဆီးနိုင်ပြီးတော့ လိုချင်တဲ့ result ကိုရရှိနိုင်ပါတယ် ။ ဒါ့အပြင် ကျွန်တော်တို့က ပျက်ဆီးနေတာကိုလည်းပြန်လည်ပြုပြင်နိုင်သွားပါတယ် ။

```
jok3r@lucy:~$ ls -l "two words.txt"
-rw-rw-r-- 1 jok3r jok3r 0 May 12 00:18 'two words.txt'
jok3r@lucy:~$ mv "two words.txt" two_words.txt
jok3r@lucy:~$
```

အဲဒီမှာ ! အခုဆိုရင်ကျွန်တော်တို့အနေနဲ့ ဒီ မကောင်းတဲ့ double quote တွေကိုဆက်ရိုက်နေစရာမလို တော့ပါဘူး ။

မှတ်ထားပါ ။ parameter expansion, arithmetic expansion နဲ့ command substitution တို့ကတော့ double quote အတွင်းမှာဆက်ပြီးအလုပ်လုပ်နေဦးမယ်ဆိုတာကိုပေါ့ ။

```
jok3r@lucy:~$ echo "$USER $((2+2)) $(cal)"
```

```
jok3r 4      May 2022
```

```
Su Mo Tu We Th Fr Sa
```

```
1 2 3 4 5 6 7
```

```
8 9 10 11 12 13 14
```

```
15 16 17 18 19 20 21
```

```
22 23 24 25 26 27 28
```

```
29 30 31
```

```
jok3r@lucy:~$
```

ကျွန်တော်တို့ခဏလောက်ရပ်ပြီးတော့ command substitution အပေါ်မှာ double quote ကဘယ်လို effect သက်ရောက်စေသလဲဆိုတာကိုတချက်လောက်ကြည့်သင့်ပါတယ် ။ ပထမအနေနဲ့ word splitting ဘယ်လိုအလုပ် လုပ်သလဲဆိုတာကိုနည်းနည်းလောက်အသေးစိတ်နက်နက်ရှိုင်းရှိုင်းကြည့်ကြစို့ ။ ကျွန်တော်တို့ရဲ့အစောပိုင်းဥပမာ တွေမှာ word splitting က ပိုနေတဲ့ whitespace တွေကိုဘယ်လိုဖယ်ရှားသလဲဆိုတာကိုတွေ့ခဲ့ကြပါတယ် ။

```
jok3r@lucy:~$ echo this is a      test
```

```
this is a test
```

```
jok3r@lucy:~$
```

default အနေနဲ့ word splitting က whitespace တွေ tab တွေ new line (linefeed character တွေ) ရှိ မရှိ လိုက်ရှာပြီးတော့သူတို့ကို စကားလုံးတွေကြားက delimiter (comma , တွေလိုစကားလုံးတွေကိုကြားဖြတ်ပေး တဲ့ character) တွေအနေနဲ့ဆက်ဆံပါတယ် ။ ဒါကဘာကိုဆိုလိုသလဲဆိုတော့ quote ၊ ခုကြားမှာရှိမနေတဲ့ (unquoted) whitespace တွေ tab တွေ new line (linefeed character)တွေ ကို စာသား text ရဲ့အစိတ်အပိုင်း တခုအနေနဲ့မစဉ်းစားပေးတော့ပါဘူး လို့ဆိုလိုတာပါပဲ ။ သူတို့တွေက separator (ကြားခြားပေးတဲ့အရာ) တွေအဖြစ် ပဲအသုံးပြုခံရပါတော့တယ် ။ သူတို့တွေက စကားလုံး word တွေကိုမတူညီတဲ့ argument တွေအဖြစ်ခွဲချလိုက်တဲ့ အတွက်ကျွန်တော်တို့ရဲ့ (အပေါ်ကဥပမာထဲက) command-line ထဲက command တခုရဲ့နောက်မှာမတူကွဲပြားတဲ့ argument ၄ ခုပါရှိနေပါတယ် ။ (this ကတခု is ကတခု a ကတခု testကတခု) တကယ်လို့ကျွန်တော်တို့က

double quote ထည့်ခဲ့မယ်ဆိုရင် ၊ ဘယ်လိုပဲဖြစ်ဖြစ် ၊ word splitting က ဖိနှိပ်ကန့်သတ်လိုက်ပြီးတော့ embedded space တွေကို delimiter အဖြစ်မဆက်ဆံတော့ပါဘူး ။ အဲဒီအစား သူတို့က argument ရဲ့ အစိတ်အပိုင်းတခုဖြစ်သွားပါတယ် ။

```
jok3r@lucy:~$ echo "this is a test"
this is a test
jok3r@lucy:~$
```

double quote တွေကိုပေါင်းထည့်လိုက်တာနဲ့ကျွန်တော်တို့ရဲ့ command-line ထဲမှာ command တခုရဲ့နောက်မှာ single argument တခုကပ်ပါလာတာ ပါဝင်လာပြီဖြစ်ပါတယ် ။ တကယ်တော့ word splitting mechanism ကနေ newline တွေကို delimiter တွေအနေနဲ့သတ်မှတ်လိုက်ခြင်းဟာ command substitution အပေါ်မှာ စိတ်ဝင်စားစရာကောင်းတဲ့ ၊ effect သက်ရောက်ဖြစ်ပေါ်စေပါတယ် ။ အောက်ကဥပမာကိုစဉ်းစားဆုံးဖြတ်ကြည့်ပါ ။

```
jok3r@lucy:~$ echo $(cal)
May 2022 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
31
jok3r@lucy:~$ echo "$(cal)"
May 2022
Su Mo Tu We Th Fr Sa
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
jok3r@lucy:~$
```

ပထမစစ်ချင်းမှာ quote မလုပ်ထားတဲ့ command substitution က argument ၃၈ လုံးပါဝင်တဲ့ command-line တခုအဖြစ် result ထွက်လာပါတယ် ။ ဒုတိယမှာတော့ result က argument ၁ လုံးပဲပါတဲ့ command-line တခုဖြစ်သွားပြီးတော့သူ့မှာ embedded space တွေနဲ့ newline တွေပါဝင်နေပါတယ် ။

Single Quotes

ကျွန်တော်တို့အနေနဲ့ expansion တွေအကုန်လုံးကိုဖိနှိပ်ကန့်သတ်ဖို့လိုအပ်လာတဲ့အခါမှာ ကျွန်တော်တို့က single quote တွေကိုအသုံးပြုပါတယ် ။ ဒီမှာ quote မလုပ်ထားတာနဲ့ double quote တွေနဲ့ single quote တွေကို နှိုင်းယှဉ်ပြထားပေးပါတယ် ။

```
jok3r@lucy:~$ echo text ~/.txt {a,b} $(echo foo) $((2+2)) $USER
text /home/jok3r/two_words.txt a b foo 4 jok3r
jok3r@lucy:~$ echo "text ~/.txt {a,b} $(echo foo) $((2+2)) $USER"
text ~/.txt {a,b} foo 4 jok3r
jok3r@lucy:~$ echo 'text ~/.txt {a,b} $(echo foo) $((2+2)) $USER'
text ~/.txt {a,b} $(echo foo) $((2+2)) $USER
jok3r@lucy:~$
```

ကျွန်တော်တို့မြင်ရတဲ့အတိုင်း quoting တွေ level တက်လာလေလေ expansion တွေဟာ ပို၍ပို၍ ဖိနှိပ်ကန့်သတ် ခံလာရလေလေ ဖြစ်ပါတယ် ။

Escaping Characters

တခါတလေမှာကျွန်တော်တို့အနေနဲ့ single character တလုံးထဲကိုပဲ quote လုပ်ချင်တာမျိုးရှိပါတယ် ။ အဲ့ ဒီလိုလုပ်ဖို့အတွက်ကျွန်တော်တို့က character တလုံးရဲ့အရှေ့မှာ backslash (\) တခုထည့်ပေးလိုက်တာ ၊ အဲ့ဒါမျိုး ကိုအခုကိစ္စမှာတော့ escape character လို့ခေါ်ပါတယ် ။ မကြာခဏဆိုသလို သူ့ကို double quote ထဲမှာထည့်ပြီး expansion တွေကိုရွေးပြီးတော့တားဆီး (ဖိနှိပ်ကန့်သတ်) တာမျိုးလုပ်ပါတယ် ။

```
jok3r@lucy:~$ echo "The balance for user $USER is: \$5.00"
The balance for user jok3r is: $5.00
jok3r@lucy:~$
```

သာမန်အားဖြင့်တော့ filename တခုထဲက character တခုရဲ့ special meaning ကိုပယ်ဖျက်ဖို့ အတွက် escaping တွေကိုသုံးပါတယ် ။ ဥပမာ - အဲ့ဒါ (escaping တွေ) ကိုပုံမှန်အားဖြင့် shell အတွက် special meaning ရှိနေတဲ့ filename တွေထဲက character တွေမှာသုံးဖို့ရနိုင်ပါတယ် ။ အဲ့ဒါတွေထဲမှာ \$, ! , & , (space တခု) နဲ့အခြားဟာတွေပါဝင်ပါလိမ့်မယ် ။ filename တခုထဲမှာ special character တခုထည့်ဖို့ဆိုရင်ခင်ဗျားအနေနဲ့ ဒီလိုလုပ်လို့ရပါတယ် ။

backslash character (\) တခုပေါ်အောင်လုပ်ချင်ရင်တော့ \\ လို့ရိုက်ပြီးတော့ escape လုပ်နိုင်ပါတယ်။ တခုမှတ်ထားရမှာက single quote အတွင်းမှာဆိုရင် backslash ကသူ့ရဲ့ special meaning ပျောက်သွားပြီးတော့ ရိုးရိုးသာမန် character တခုအနေနဲ့ပဲဆက်ဆံခံရမှာဖြစ်ပါတယ်။

BACKSLASH ESCAPE SEQUENCES

နောက်တခုအနေနဲ့ သူက escape character အနေနဲ့ပေါ့ ၊ backslash ကို control code တွေလိုမျိုး သတ်မှတ်ထားတဲ့ special character တွေအနေနဲ့ကိုယ်စားပြုဖို့အတွက် notation ရေးတဲ့အခါမှာတွဲသုံးပါတယ်။ ASCII coding scheme ထဲက ပထမဦးဆုံးသော character ၃၂ လုံးကို teletype-like device တွေဆီ command တွေထုတ်လွှင့်ပေးပို့တဲ့အခါမှာသုံးပါတယ်။ အဲဒီထဲကတချို့ code တွေ (ab, backspace, linefeed နဲ့ carriage return) ကတော့ရင်းနှီးပြီးသားဖြစ်ကြပေမယ့် အခြား (null, end-of-transmission နဲ့ acknowledge) လိုဟာမျိုး တွေကျတော့မရင်းနှီးသေးပါဘူး။ Table 7-2 ထဲမှာပြထားတဲ့အတိုင်းပါပဲ။

Table 7-2: Backslash Escape Sequences

Escape Sequence	Meaning
\a	Bell ပါ (“alert”— computer ကို beep ဆိုပြီးအသံမြည်စေပါတယ်)
\b	Backspace ပါ
\n	Newline လိုင်းအသစ်ယူပေးပါတယ် (Unix-like system တွေပေါ်မှာဒါက linefeed တခု ထုတ်ပေးပါတယ်)
\r	Carriage return ပါ (စာတကြောင်းဆုံးတဲ့အခါ newline ဆီကို cursor ရောက်သွားတာကို ပြောတာပါ)
\t	Tab ပါ

ဒီ Table မှာအသုံးများတဲ့ backslash escape sequence အချို့ကို list လုပ်ပြထားပါတယ်။ ဒီ backslash ကိုသုံးပြီးလုပ်တဲ့နောက်ကွယ်ကစိတ်ကူးကို C programming ကနေစတင်ခဲ့တာဖြစ်ပြီးတော့ shell အပါအဝင်အခြား အရာတွေအများအပြားကလည်းမွေးစားယူသုံးကြပါတယ်။

echo မှာ -e option ကိုထည့်ခြင်းဖြင့် escape sequence တွေကို interpretation လုပ်ခွင့်ပြုပေးပါတယ်။ ။
ခင်ဗျားအနေနဲ့သူတို့ကို '\$' ' အတွင်းမှာထည့်လည်းရပါတယ်။ ။ ဒီမှာ sleep command (သတ်မှတ်ထားတဲ့ စက္ကန့်
နံပါတ်ကိုစောင့်ပေးပြီးတော့ exit လုပ်ပေးတဲ့ ရိုးရှင်းတဲ့ program တခု) ကိုသုံးပြီးတော့ ကျွန်တော်တို့က အခြေခံကျ
တဲ့ countdown timer တခုကိုဖန်တီးကြည့်ပါမယ်။ ။

```
sleep 10; echo -e "Time's up\a"
```

ကျွန်တော်တို့က ဒါကို ဒီလိုလုပ်လည်းရပါတယ်။ ။

```
sleep 10; echo "Time's up" $\a'
```

Final Note

ကျွန်တော်တို့အနေနဲ့ shell ကိုအသုံးပြုပြီးတော့ရှေ့ဆက်နေစဉ်မှာ expansion နဲ့ quoting အသုံးပြုတာ
တွေကမကြာခဏဆက်တိုက်ဆိုသလိုတိုးလာတာကိုကျွန်တော်တို့တွေ့လာရမှာဖြစ်ပါတယ်။ ။ ဒါကြောင့် သူတို့တတွေ
ရဲ့အလုပ်လုပ်ပုံတွေကိုကောင်းကောင်းနားလည်ထားသင့်တာကအဓိပ္ပါယ်ရှိပါတယ်။ ။ အမှန်တော့သူတို့ဟာ shell
လေ့လာသင်ကြားရာမှာအရေးအကြီးဆုံးအကြောင်းအရာဖြစ်တယ်ဆိုတာကိုတော့အငြင်းပွားဖွယ်ရာရှိပါတယ်။ ။
expansion ကိုသေသေချာချာနားမလည်ခဲ့ရင်တော့ shell ဆိုတဲ့အရာဟာ ဆန်းကြယ်ပြီးစိတ်ရှုပ်ထွေးစရာတို့ရဲ့
အကြောင်းအရင်းဖြစ်လာပြီးတော့သူ့ရဲ့ potential power စွမ်းအားတွေကိုလည်း ဖြုန်းတီးပစ်သလို ဖြစ်နေပါလိမ့်
မယ်။ ။

8

ADVANCED KEYBOARD

TRICKS

ကျွန်တော်တခါတလေမှာ စနောက်ပြီးတော့ Unix ကို "စာရိုက်ရတာကြိုက်တဲ့လူတွေအတွက် Operation System " လို့ပြောလေ့ရှိပါတယ် ။ ဟုတ်ပါတယ် ။ အမှန်ကတော့ သူ့မှာ command-line ပါဝင်နေတာကိုကအဲ့ဒီဟာ ကိုသက်သေအထောက်အထားပြနေသလိုဖြစ်နေတာပါ ။ ဒါပေမယ့် command-line အသုံးပြုသူတွေက စာရိုက်ရတာကို ဒီလောက်ကြီးလည်း ကြိုက်မနေကြပါဘူး ။ ဘာလို့မြောက်မြားစွာသော command တွေမှာ cp , ls , mv နဲ့ rm လိုမျိုး shortcut တွေရှိနေရတာလဲ ?

အမှန်ကတော့ command-line ရဲ့တန်ဖိုးထားစရာအကောင်းဆုံးပန်းတိုင်တွေထဲကတခုကတော့ အပျင်းတက်ခြင်း ----- စာရိုက်ချက်အနည်းဆုံးနဲ့အလုပ်တော်တော်များများကိုလုပ်နိုင်ဖို့ပါပဲ ။ XD ။ နောက်ထပ်ပန်းတိုင်တခုတော့ခင်ဗျားရဲ့လက်ချောင်းလေးတွေကို keyboard ကနေမခွာဖို့ ---- mouse ကိုလှမ်းမကိုင်ဖို့ပါပဲ ။ အခု Chapter မှာကျွန်တော်တို့က Keyboard ကိုအသုံးပြုရာမှာပိုမိုမြန်ဆန်စေပြီးတော့ပိုမိုထိရောက်စေတဲ့ bash feature တွေကိုလေ့လာကြမှာဖြစ်ပါတယ် ။

အောက်ဖော်ပြပါ command တွေပါဝင်လာမှာဖြစ်ပါတယ် ။

- **Clear ----** (terminal ထဲက) screen ပေါ်ကစာတွေအကုန်ရှင်းပေးပါတယ် ။
- **history ----** history list ထဲက content တွေကိုဖော်ပြပေးပါတယ် ။

Command Line Editing

bash က command line editing ကို implement ပြုလုပ်ဖို့အတွက် Readline လို့ခေါ်တဲ့ library တခု ((library ဆိုတာ) မတူညီတဲ့ program တွေကယူသုံးနေပြီး(အားလုံး) မျှဝေသုံးစွဲနေတဲ့လမ်းကြောင်း အစုအဝေးတခု) ကိုအသုံးပြုပါတယ် ။ ကျွန်တော်တို့အနေနဲ့ဒါမျိုးအချို့ကိုတွေ့မြင်ခဲ့ပြီးသားဖြစ်ပါတယ် ။ ကျွန်တော် တို့သိပါတယ် ။ ဥပမာ - arrow key က cursor ကိုရွှေ့စေနိုင်ပါတယ် ။ ဒါပေမယ့်အဲ့ဒီမှာနောက်ထပ်များစွာသော feature တွေရှိနေပါသေးတယ် ။ ဒါတွေကိုကျွန်တော်တို့အလုပ်ထဲမှာအသုံးချလို့ရတဲ့ထပ်တိုး tools တွေအနေနဲ့တွေး ကြည့်လိုက်ပါ ။ ဒါတွေအကုန်လုံးကိုတတ်မြောက်အောင်သင်ကြားဖို့အရေးကြီးပါဘူး ။ ဒါပေမယ့်သူတို့ထဲကအများ စုကတော့အရမ်းအသုံးဝင်ပါတယ် ။ မိမိနှစ်သက်ရာကိုကောက်ယူရွေးချယ်လိုက်ပါ ။

Note: အောက်မှာပြထားတဲ့ key sequence တွေအချို့ (သာမန်အားဖြင့်တော့ Alt key နဲ့တွဲသုံးတဲ့ ကောင်တွေ) ကို GUI ကအခြား function တွေအတွက်ကြားဖြတ်ယူတာမျိုးလည်းရှိ တတ်ပါတယ် ။ key sequence တွေအကုန်လုံးက virtual console (Linux console သို့မဟုတ် terminal လို့လည်း ခေါ်ကြပါတယ် ။ အဓိပ္ပါယ်ကွဲလွဲမှုအနည်းငယ်သာရှိပေမယ့် အတူတူလိုပါပဲ ။) တခု ပေါ်မှာတော့ ကောင်းမွန်စွာအလုပ်လုပ်ပါတယ် ။

Cursor Movement

Table 8-1 မှာ cursor ကိုရွှေ့ဖို့သုံးတဲ့ key တွေကိုပြထားပေးပါတယ် ။

Table 8-1: Cursor Movement Commands

Key	Action
CTRL -A	cursor ကို line ရဲ့အစဆီကိုပို့ပေးပါတယ် ။
CTRL -E	cursor ကို line ရဲ့အဆုံးဆီကိုပို့ပေးပါတယ် ။
CTRL -F	cursor ကို character တလုံးစာအရှေ့သို့ (ညာဖက်သို့) ပို့ပေးပါတယ် ။ right arrow key နဲ့ အတူတူပါပဲ ။
CTRL -B	cursor ကို character တလုံးစာအနောက်သို့ (ဗယ်ဖက်သို့) ပို့ပေးပါတယ် ။ left arrow key နဲ့ အတူတူပါပဲ ။
ALT -F	cursor ကို စကားစုတခု word ရဲ့အဆုံး(ညာဖက်)ဆီကိုပို့ပေးပါတယ် ။
ALT -B	cursor ကို စကားစုတခု word ရဲ့အစ(ဗယ်ဖက်)ဆီကိုပို့ပေးပါတယ် ။
ALT -L	screen ကိုရှင်းပေးပြီးတော့ cursor ကို(screen ရဲ့)အပေါ်ဗယ်ဖက်ထောင့်ကိုပို့ပေးပါတယ် ။ clear command ကလည်းသူ့လိုပဲအလုပ်လုပ်ပေးပါတယ် ။

Modifying Text

Table 8-2 မှာ command-line ပေါ်မှာ character တွေကို edit လုပ်တဲ့ keyboard command တွေကို list လုပ်ပြထားပါတယ် ။

Cutting and Pasting (Killing and Yanking) Text

Readline documentation မှာ ကျွန်တော်တို့အနေနဲ့ပုံမှန်အားဖြင့်ခေါ်ဝေါ်သုံးစွဲနေကြ cut လုပ်ခြင်းနဲ့ paste လုပ်ခြင်းတွေကိုသူက kill လုပ်ခြင်းနဲ့ yank လုပ်ခြင်းဆိုပြီးခေါ်ဝေါ်သုံးစွဲပါတယ် ။ Table 8-3 မှာ cut လုပ်ခြင်းနဲ့ paste လုပ်ခြင်းတွေအတွက် command တွေကို list လုပ်ပြထားပါတယ် ။ cut လုပ်လိုက်တဲ့အရာ (item) တွေကို kill-ring လို့ခေါ်တဲ့ buffer ထဲမှာသွားသိမ်းထားပါတယ် ။

Table 8-2: Text Editing Commands

Key	Action
CTRL -D	cursor ရောက်နေတဲ့နေရာက character တလုံးကိုဖျက်ပေးပါတယ် ။
CTRL -T	cursor ရောက်နေတဲ့နေရာက character တလုံးနဲ့သူ့အရှေ့ (ဗယ်ဖက်)က character တလုံး နဲ့ transpose နေရာချင်းလဲပေးပါတယ် ။ (ဆက်နှိပ်ရင်ညာဖက်က character တလုံးဆီနဲ့ဆက်တိုက်နေရာလဲပေးသွားမှာပါ ။ တကယ်လို့ cursor ကအရှေ့ဆုံးမှာရောက်နေရင်ဘာမှာလုပ်ပေးမှာမဟုတ်ပါဘူး ။ ဒုတိယ character မှာ cursor ကိုချထားပြီးလုပ်မှရပါမယ် ။ ဥပမာ - Hello မှာဆိုရင် e နေရာမှာ cursor ချထားပါ ။)
ALT -T	cursor ရောက်နေတဲ့နေရာက word စကားလုံးတခုနဲ့သူ့အရှေ့ (ဗယ်ဖက်)က word စကားလုံးတခု နဲ့ transpose နေရာချင်းလဲပေးပါတယ် ။ (ဆက်နှိပ်ရင်ညာဖက်က word စကားလုံးတခုဆီနဲ့ဆက်တိုက်နေရာလဲပေးသွားမှာပါ ။ တကယ်လို့ cursor ကအရှေ့ဆုံးမှာရောက်နေရင်ဘာမှာလုပ်ပေးမှာမဟုတ်ပါဘူး ။ ဒုတိယ word စကားလုံးတခု cursor ကိုချထားပြီးလုပ်မှရပါမယ် ။ ဥပမာ - Hello World မှာဆိုရင် ဒုတိယစကားလုံးဖြစ်တဲ့ World နေရာမှာ cursor ချထားပါ ။)
ALT -L	cursor ရောက်နေတဲ့နေရာကနေ word စကားလုံးတခုရဲ့(ဗယ်ဖက်)အဆုံးအထိ စာလုံးအသေး lowercase တွေအဖြစ်ပြောင်းပေးမှာဖြစ်ပါတယ် ။
ALT -U	cursor ရောက်နေတဲ့နေရာကနေ word စကားလုံးတခုရဲ့(ဗယ်ဖက်)အဆုံးအထိ စာလုံးအသေး Uppercase တွေအဖြစ်ပြောင်းပေးမှာဖြစ်ပါတယ် ။

able 8-3: Cut and Paste Commands

Key	Action
CTRL -K	cursor ရောက်နေတဲ့နေရာကနေ (ညာဖက်) စာကြောင်းအဆုံးအထိရှိသမျှစာတွေအကုန် kill ဖျက်ပေးမှာဖြစ်ပါတယ် ။ (cursor ကစာကြောင်းရဲ့နောက်ဆုံးရောက်နေမယ်ဆိုရင်အလုပ်လုပ်ပေးမှာမဟုတ်ပါဘူး ။)
CTRL -U	cursor ရောက်နေတဲ့နေရာကနေ (ဗယ်ဖက်) စာကြောင်းအရှေ့ဆုံးအထိရှိသမျှစာတွေအကုန် kill ဖျက်ပေးမှာဖြစ်ပါတယ် ။ (cursor ကစာကြောင်းရဲ့ရှေ့ဆုံးရောက်နေမယ်ဆိုရင်အလုပ်လုပ်ပေးမှာမဟုတ်ပါဘူး ။)
ALT -D	cursor ရောက်နေတဲ့နေရာကနေ (ညာဖက်) စကားလုံးတလုံးစာ word အဆုံးအထိရှိသမျှစာတွေအကုန် kill ဖျက်ပေးမှာဖြစ်ပါတယ် ။ (cursor ကစာကြောင်းရဲ့နောက်ဆုံးရောက်နေမယ်ဆိုရင်အလုပ်လုပ်ပေးမှာမဟုတ်ပါဘူး ။)
ALT -BACK SPACE	cursor ရောက်နေတဲ့နေရာကနေ (ဗယ်ဖက်) စာကြောင်းအရှေ့ဆုံးအထိရှိသမျှစာတွေအကုန် kill ဖျက်ပေးမှာဖြစ်ပါတယ် ။ တကယ်လို့ cursor ကစာကြောင်းရဲ့ရှေ့ဆုံးကစကားလုံးရဲ့အရှေ့ဆုံးမှာရောက်နေမယ်ဆိုရင် အလုပ်လုပ်ပေးမှာမဟုတ်ပါဘူး။)
CTRL -Y	cursor ရောက်နေတဲ့နေရာမှာ kill-ring (memory buffer) ထဲကနေ yank (paste ချတာ) လုပ်ပေးပါတယ် ။

THE META KEY

ခင်ဗျားအနေနဲ့ bash man page က “READLINE” section ထဲမှာတွေ့နိုင်တဲ့ Readline documentation ထဲကိုစွန့်စားခရီးလေးထွက်ခဲ့မယ်ဆိုရင် ခင်ဗျားအနေနဲ့ meta key ဆိုတဲ့အသုံးအနှုန်းနဲ့တွေ့ကြုံရမှာပဲဖြစ်ပါတယ် ။ ခေတ်သစ် keyboard တွေပေါ်မှာဆိုရင်တော့ဒါက Alt key ကိုညွှန်းပေးမယ့်လည်းဒါကအမြဲတမ်းတော့မဟုတ်ပါဘူး ။

အရင်တုန်းကမပွင့်လင်းခင်အချိန် (Pc တွေမပေါ်ခင် Unix ပေါ်ထွက်ပြီးချိန်) တွေတုန်းကလူတိုင်းမှာကိုယ်ပိုင် computer တွေမရှိကြသေးပါဘူး ။ သူတို့မှာရှိကောင်းရှိနိုင်ခဲ့မယ့် device ကတော့ terminal တခုပဲဖြစ်ပါတယ် ။ terminal ဆိုတာကတော့ စာသားတွေချည်းပဲဖော်ပြနိုင်တဲ့ text display screen တခုနဲ့ keyboard တခုပါရှိတဲ့ဆက်သွယ်ရေးကိရိယာတခုဖြစ်ပြီးတော့ သူ့မှာ cursor ကိုဟိုဟိုဒီဒီရွှေ့ဖို့နဲ့ text character တွေကိုဖော်ပြနိုင်ဖို့အတွက်လုံလောက်တဲ့ Electronic (စက်ကိရိယာ) တွေပါဝင်ပါတယ် ။ သူ့ကိုပိုကြီးတဲ့ computer တလုံး သို့မဟုတ်ပိုကြီးတဲ့ computer တလုံးရဲ့ဆက်သွယ်ရေးကွန်ယက်တခုဆီကို (ပုံမှန်အားဖြင့် serial cable နဲ့) ချိတ်ဆက်ထားပါတယ် ။ အဲဒီမှာမတူညီတဲ့ Brand (ကုန်အမှတ်တံဆိပ်) နဲ့ terminal တွေရှိပြီးတော့သူတို့အားလုံးကိုယ်စီမှာလည်း မတူညီတဲ့ Keyboard နဲ့ display feature set တွေရှိနေကြပါတယ် ။ သူတို့အားလုံးကအနည်းဆုံး ASCII ကိုနားလည်

တယ်ဆိုကထဲက developer တွေအနေနဲ့ အနိမ့်ဆုံး သာမန်ရဲ့တဝက်လောက်ပဲရှိတဲ့ portable application တွေကို ရေးချင်ခဲ့ကြပါတယ် ။ Unix system တွေမှာ terminal တွေနဲ့သူတို့ရဲ့ display feature တွေအတွက် အလွန် ဂရုတစိုက်အစီအစဉ်တကျကိုင်တွယ်လုပ်ဆောင်နိုင်တဲ့နည်းလမ်းတခုရှိပါတယ် ။ Readline ကိုရေးတဲ့ developer တွေမှာ သတ်မှတ်ထားတဲ့အပို control key သေချာမရှိသေးတာကြောင့် သူတို့အနေနဲ့တခုတီထွင်လိုက်ပြီးတော့အဲ့ဒါ ကို meta လို့နာမည်ပေးလိုက်ပါတယ် ။ ခေတ်သစ် Keyboard တွေပေါ်မှာ Alt key က meta အနေနဲ့ဆောင်ရွက်ပေး တာကြောင့် ခင်ဗျားအနေနဲ့ Esc key (Escape key) ကိုနှိပ်ပြီးပြန်လွှတ်လိုက်ခြင်းအားဖြင့် Alt key ကိုဖိထားတာနဲ့ တူညီတဲ့ effect ကို တကယ်လို့ခင်ဗျားက terminal ကိုသုံးနေသေးတယ်ဆိုရင် (Linux ထဲမှာအခုထိသုံးလို့ရနေသေး တယ်) ရရှိနိုင်မှာဖြစ်ပါတယ် ။

Completion

shell က mechanism တခုကနေတဆင့်ခင်ဗျားကိုအကူအညီပေးနိုင်တဲ့နောက်ထပ်နည်းလမ်းတခုကတော့ completion ဖြစ်ပါတယ် ။ command တခုကိုရိုက်နေတုန်းမှာ TAB ကိုနှိပ်လိုက်ရင် completion ဖြစ်သွားပါတယ် ။ သူဘယ်လိုအလုပ်လုပ်သလဲကြည့်ကြစို့ ။ ခင်ဗျားရဲ့ home directory က ဒီလိုပုံစံဖြစ်နေမယ်ဆိုပါစို့ ။

```
jok3r@lucy:~$ ls
default.ovpn  hacking_tutorial_link  Music      Templates
Desktop       javasharedresources   Pictures   Videos
Documents     ls-output.txt         pt
Downloads     master.zip            Public
Dropbox       MEGAsync              Recordings
dwhelper      mm-kb-master          snap
jok3r@lucy:~$
```

အောက်ကဟာကိုရိုက်ကြည့်လိုက်ပါ ဒါပေမယ့် Enter key ကိုမနှိပ်ပါနဲ့ ။

```
jok3r@lucy:~$ ls l
```

အခု TAB key ကိုနှိပ်လိုက်ပါ

jok3r@lucy:~\$ ls ls-output.txt

shell ကခင်ဗျားအတွက် line ကိုဘယ်လို complete ဖြစ်အောင်လုပ်ပေးသွားသလဲဆိုတာကိုတွေ့လား ? နောက်တခုထပ်လုပ်ကြည့်ကြစို့။ ။ ထပ်ပြီးတော့ ၊ Enter key ကိုမနှိပ်ပါနဲ့။ ။

jok3r@lucy:~\$ ls D

TAB နှိပ်လိုက်ပါ

jok3r@lucy:~\$ ls D

completion မဖြစ်ပဲ Beep ဆိုပြီးအသံမြည်လာပါလိမ့်မယ်။ ။ ဒီလိုဖြစ်တာဘာကြောင့်လဲဆိုတော့ D က (D နဲ့စတာတွေက) directory ထဲမှာတခုထက်ပိုပြီးရှိနေလို့ပါပဲ။ ။ completion အောင်မြင်ဖို့အတွက်ဆိုရင် ခင်ဗျားအနေနဲ့သူ့ကို (shell ကို) အစဖော်ပေးတဲ့ဟာက သိသာမြင်သာစေရမှာဖြစ်ပါတယ်။ ။ (ကိုယ့်ထဲမှာရှိတဲ့ directory သို့မဟုတ် file တွေရဲ့နာမည်ကိုပဲရိုက်ရင်အဆင်ပြေပါတယ်။) ကျွန်တော်တို့အနေနဲ့ဆက်သွားလို့ရပါတယ်။ ။

jok3r@lucy:~\$ ls Do

TAB နှိပ်လိုက်ပါ

jok3r@lucy:~\$ ls Documents

completion အောင်မြင်သွားပါပြီ။ ။

အခု ဥပမာ က completion မှာအသုံးအများဆုံးလည်းဖြစ်တဲ့ pathname တွေရဲ့ completion ကိုပြထားပေးမယ့် completion က variable (တကယ်လို့စကားလုံး word ရဲ့အစမှာ \$ ပါလာခဲ့ရင်) ၊ username (တကယ်လို့စကားလုံး word ရဲ့အစမှာ ~ ပါလာခဲ့ရင်) ၊ command (တကယ်လို့စကားလုံး word ရဲ့အစမှာ word ပါလာခဲ့ရင်)

ပြီးတော့ hostname (တကယ်လို့စကားလုံး word ရဲ့အစမှာ @ ပါလာခဲ့ရင်) တွေအပေါ်မှာလည်းအလုပ်လုပ်ပေးပါတယ်။ Hostname completion က /etc/host ထဲက list ထဲမှာပါတဲ့ hostname တွေအတွက်သာအလုပ်လုပ်ပေးပါတယ်။

control နဲ့ meta key sequence အနည်းငယ်သာ completion နဲ့တွဲဖက်ဆက်စပ်နေပါတယ်။ (Table 8-4 မှာကြည့်ပါ။)

Table 8-4: Completion Commands

Key	Action
ALT - ?	ဖြစ်နိုင်ခြေရှိတဲ့ completion တွေကိုဖော်ပြပေးပါတယ်။ system အများစုပေါ်မှာတော့ဒါကိုခင်ဗျားအနေနဲ့ TAB key ၊ ချက်နှိပ်ပြီးတော့လုပ်ဆောင်နိုင်ပါတယ်။ အဲဒါကပိုပြီးလွယ်ကူစေပါတယ်။
ALT - *	ဖြစ်နိုင်ခြေရှိတဲ့ completion တွေကိုအကုန်လုံးကိုထည့်ပေးပါတယ်။ ခင်ဗျားအနေနဲ့ တခုထက်ပိုတဲ့ဖြစ်နိုင်ခြေတွေနဲ့တိုက်ဆိုင်ချင်တဲ့အခါမျိုးမှာအဲဒါကအသုံးဝင်ပါတယ်။

အဲဒီမှာကျွန်တော်ရှာဖွေနိုင်တာထက်ပိုနည်းတယ်ဆိုတာထက်ပိုပြီးမှေးမှိန်နေပါတယ်။ ခင်ဗျားအနေနဲ့ bash man page ထဲက “READLINE” section အောက်မှာ list တခုကိုတွေ့နိုင်ပါတယ်။

PROGRAMMABLE COMPLETION

bash ရဲ့လက်ရှိ version မှာ programmable completion လို့ခေါ်တဲ့ facility တခုရှိပါတယ်။ programmable completion က (သို့မဟုတ် ၊ ပိုပြီးဖြစ်နိုင်တာက ၊ ခင်ဗျားရဲ့ distribution provider က) ခင်ဗျားကိုနောက်ထပ် completion rule တွေထပ်ထည့်ခွင့်ပြုပါတယ်။ ပုံမှန်အားဖြင့်ဒါမျိုးကို သတ်မှတ်ထားတဲ့ application အတွက် support ပေါင်းထည့်ချင်တဲ့အခါမှာလုပ်လေ့ရှိပါတယ်။ ဥပမာ - ဒါမျိုးကို command တခုရဲ့ option list အတွက် completion ထပ်ထည့်ချင်တာ သို့မဟုတ် application တခုကနေ support ပေးတဲ့သက်ဆိုင်ရာ file type တွေနဲ့ match တိုက်ဆိုင်စစ်ဆေးတဲ့အခါမှာဒါမျိုးကဖြစ်နိုင်ပါတယ်။ Ubuntu မှာဆိုရင် default အနေနဲ့သတ်မှတ်ထားတဲ့ အတော်အသင့်ကြီးမားတဲ့ set တခုရှိပါတယ်။ Programmable completion ကို shell function တွေက implement လုပ်ပေးပါတယ်။ ကျွန်တော်တို့နောက်ပိုင်း chapter တွေကျရင်ဆက်လေ့လာသွားမယ့် shell script အသေးစားလေးတခုဖြစ်ပါတယ်။ တကယ်လို့ခင်ဗျားစပ်စုချင်တယ်ဆိုရင် ဒါကိုလုပ်ကြည့်ပါ။

set | less

ပြီးတော့ခင်ဗျားရှာတွေ့နိုင်မလားကြည့်လိုက်ပါ။ distribution တွေအကုန်လုံးမှာတော့သူက default အနေနဲ့ပါမလာတတ်ပါဘူး။

Using History

Chapter 1 မှာကျွန်တော်တို့ရှာဖွေသိရှိခဲ့တဲ့အတိုင်း bash က ရိုက်ထည့်လိုက်ပြီးသား command တွေရဲ့ history ကိုထိမ်းသိမ်း (မှတ်) ထားပေးပါတယ်။ အဲဒီ command တွေရဲ့ list ကို home directory ထဲက .bash_history ဆိုတဲ့ file တခုထဲမှာထားထားပါတယ်။ history facility ဟာ ခင်ဗျားလုပ်ရမယ့်စာရိုက်ခြင်းအရေအတွက်ကိုလျှော့ချပေးတဲ့နေရာမှာအသုံးဝင်တဲ့အရင်းအမြစ်တခုဖြစ်ပါတယ်။ အထူးသဖြင့် command-line editing နဲ့ပေါင်းစပ်လုပ်ကိုင်ရတဲ့အခါမှာပေါ့။

Searching History

ဘယ်အချိန်မှာမဆို ကျွန်တော်တို့အနေနဲ့ history list ထဲက content တွေကိုကြည့်လို့ရပါတယ်။

```
jok3r@lucy:~$ history | less
```

default အနေနဲ့ bash ကခင်ဗျားရိုက်ပြီးသား command အခုရေ 500 ကိုသိမ်းထားပေးပါတယ်။ ကျွန်တော်တို့အနေနဲ့ဒါကိုဘယ်လို adjust လုပ်ရတယ်ဆိုတာကို Chapter 11 ကျရင်တွေ့ပါလိမ့်မယ်။ ကျွန်တော်တို့က /usr/bin ကို list လုပ်ဖို့အတွက်သုံးခဲ့တဲ့ command ကို ကျွန်တော်တို့ကပြန်ရှာချင်တယ်ဆိုကြပါစို့။ ဒါကိုလုပ်နိုင်တဲ့နည်းတခုနည်းကဒီမှာပါ။

```
jok3r@lucy:~$ history | grep /usr/bin
```

ပြီးတော့ကျွန်တော်တို့ရတဲ့ result တွေထဲမှာမှ စိတ်ဝင်စားစရာကောင်းတဲ့ဒီလို command မျိုးပါတဲ့ line တခုကို ကျွန်တော်တို့က ရခဲ့တယ်ဆိုပါစို့။

```
1755 ls -l /usr/bin > ls-output.txt
```

နံပါတ် 1755 ဆိုတာ history ထဲက command ရဲ့ line number ဖြစ်ပါတယ်။ အဲဒါကိုကျွန်တော်တို့ကချက်ချင်းပဲ history expansion လို့ခေါ်တဲ့ နောက်ထပ် expansion အမျိုးအစားတခုနဲ့အသုံးပြုလို့ရပါတယ်။ ကျွန်တော်တို့အနေနဲ့ဒီလိုလုပ်လို့ရပါတယ်။

```
jok3r@lucy:~$ !1755
```

bash က !1755 ကို history list ထဲက 1755 line မြောက် content ထဲကို expand လုပ်သွားမှာဖြစ်ပါတယ်။ ကျွန်တော်တို့အနေနဲ့ history expansion ရဲ့အခြားပုံစံတွေကိုခဏကြာမှလေ့လာသွားပါမယ်။

bash မှာ history list တွေကို increment တိုးတိုးပြီးတော့ ရှာပေးနိုင်တဲ့စွမ်းရည်ထည့်သွင်းပေးထားပါတယ်။ ဒါက ဘာကိုဆိုလိုတာလဲဆိုတော့ကျွန်တော်တို့က character တွေထည့်လိုက်တဲ့အခါမှာ history list မှာသွားရှာပေးဖို့ bash ကို ကျွန်တော်တို့ကပြောလို့ရပြီး (ရိုက်ထည့်တဲ့) character တွေထပ်တိုးလာတိုင်းမှာ ကျွန်တော်တို့ရဲ့ရှာဖွေမှုကပိုကောင်းလာမှာ ဖြစ်တယ်လို့ဆိုလိုတာပါ။ increment နဲ့ရှာဖွေတာကိုစတင်ဖို့အတွက်ဆိုရင် CTRL - R နှိပ်လိုက်ပြီးတော့ကိုယ်ရှာချင်တဲ့စာသား ကိုရိုက်ထည့်ရမှာဖြစ်ပါတယ်။ ခင်ဗျားရှာတွေ့တဲ့အခါ ခင်ဗျားအနေနဲ့ Enter ရိုက်ပြီးတော့ command ကို execute လုပ်တာ သို့မဟုတ် CTRL - J နှိပ်ပြီးတော့ (TAB ကိုနှိပ်လည်းရတယ်) history list ထဲက line ကို လက်ရှိ command-line ထဲကို copy လုပ်နိုင်ပါတယ်။ နောက်ထပ်ပေါ်လာမယ့် text စာသားကိုရှာဖို့ (history list ထဲမှာ အပေါ်ကိုတက်သွားတာ) CTRL -R ကို နောက်တကြိမ်နှိပ်ပါ။ ရှာတာကနေထွက်ဖို့ဆိုရင် CTRL -G သို့မဟုတ် CTRL -C ၂ခုထဲကတခုကိုနှိပ်ပါ။ ဒီမှာကျွန်တော်တို့ လက်တွေ့မြင်နေရပါတယ်။

```
jok3r@lucy:~$
```

```
CTRL -R ကိုနှိပ်ပါ။
```

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

prompt ပြောင်းသွားရတဲ့သဘောကကျွန်တော်တို့က reverse incremental search တိုးရှာတာကိုပြောင်းပြန်လုပ် မယ် လို့ညွှန်ပြလိုက်တာဖြစ်ပါတယ်။ အဲဒါကပြောင်းပြန်ဖြစ်နေရတာဘာလို့လဲဆိုတော့ကျွန်တော်တို့ကအခုလက်ရှိကနေ အတိတ်ဆီကိုပြန်သွားရှာမှာမို့လို့ပါပဲ။ ပြီးရင်ကျွန်တော်တို့ရှာမယ့် စာသားကိုရိုက်ပါမယ်။ အခု ဥပမာမှာတော့ /usr/bin ဖြစ် ပါတယ်။

```
(reverse-i-search) `/usr/bin': ls -l /usr/bin > ls-output.txt
```

ချက်ချင်းဆိုသလိုပဲ ရှာဖွေမှုကသူ့ရဲ့ result ကိုပြန်ပေးလာပါတယ်။ အခုဆိုရင်ကျွန်တော်က Enter နှိပ်ပြီးတော့ ကျွန်တော်တို့ ရဲ့ command ကို execute လုပ်တာ ဒါမှမဟုတ် ကျွန်တော်တို့ CTRL -J နှိပ်လိုက်ပြီးတော့ command ကို နောက်ထပ်

editing တွေလုပ်ဖို့အတွက်လက်ရှိ command-line ဆီကို copy လုပ်နိုင်ပါပြီ။ copy လုပ်ကြစို့။ CTRL -J နှိပ်လိုက်ပါ။ (TAB တချက်နှိပ်လိုက်လည်းရပါတယ်။)

```
jok3r@lucy:~$ ls -l /usr/bin > ls-output.txt
```

ကျွန်တော်တို့ရဲ့ shell prompt ပြန်ရောက်လာပြီးတော့ကျွန်တော်တို့ရဲ့ command လည်း load ခေါ်ထားပြီးတော့အလုပ်လုပ်ဖို့ အဆင်သင့်ဖြစ်နေပါပြီ။

Table 8-5 မှာ history list ကို ကြိုးကိုင်ခြယ်လှယ်ထားတဲ့ Keystroke အချို့ကို list ထုတ်ပြထားပါတယ်။

Table 8-5: History Commands

Key	Action
CTRL -P	ရှေ့ကထည့်ခဲ့တဲ့ history entry ဆီကိုသွားပေးပါတယ်။ up arrow key နဲ့အတူတူပါပဲ။
CTRL -N	နောက်ထပ်ပေါ်မယ့် history entry ဆီကိုသွားပေးပါတယ်။ down arrow key နဲ့အတူတူပါပဲ။
ALT -<	history list ရဲ့အစဦးဆုံး (ထိပ်ဆုံး) ဆီကိုသွားပေးပါတယ်။
ALT ->	history list ရဲ့အနောက်ဆုံး (အောက်ဆုံး) ဆီကိုသွားပေးပါတယ်။ အဲ့ဒါက လက်ရှိ command-line ကို ပြောတာပါပဲ။
CTRL -R	Reverse incremental search ပြောင်းပြန်တိုးရှာပေးပါတယ်။ ရှာဖွေမှုတွေက လက်ရှိ command-line ကနေ အပေါ်က history list ဆီကိုတက် တက်သွားပါတယ်။
ALT -P	ပြောင်းပြန်ရှာပေးပါတယ်။ မတိုးသွားပေးပါဘူး။ ဒီ key နဲ့ဆိုရင် ရှာချင်တဲ့ string စာသားတွေကိုရိုက် ထည့်လိုက်ပြီးတော့ ရှာဖွေမှုမစတင်ခင်မှာ Enter နှိပ်ရပါမယ်။
ALT -N	ရှေ့ဆက်ရှာပေးပါတယ်။ မတိုးသွားပေးပါဘူး။
CTRL -O	history list ထဲကလက်ရှိ item ကို execute လုပ်ပေးပြီးတော့နောက်တခုဆီကိုဆက်တက်သွားပေးပါတယ်။ history list ထဲက command အတွဲလိုက်တွေကိုပြန်ပြီးတော့ execute လုပ်တဲ့အခါမျိုးမှာဒါက အသုံးဝင်ပါတယ်။

History Expansion

shell က history list ထဲက item တွေအတွက် အထူးပြုလုပ်ထားတဲ့ expansion အမျိုးအစားတခုကို

! character ကိုအသုံးပြုပြီးတော့ကမ်းလှမ်းပါတယ်။ ။ ကျွန်တော်တို့အနေနဲ့ exclamation point (!) အနောက်မှာ နံပါတ်တခုထည့်ပြီးတော့ history list ထဲကို entry တခုဘယ်လိုထည့်ခဲ့သလဲဆိုတာကိုမြင်ခဲ့ကြပြီးဖြစ်ပါတယ်။ ။ အဲဒီမှာ အခြား expansion feature တွေအရေအတွက်တခုအထိရှိနေပါသေးတယ်။ ။ (Table 8-6 ကိုကြည့်ပါ)

ကျွန်တော့်အနေနဲ့ !string နဲ့ !?string form တွေသုံးတာကိုဆန့်ကျင်သတိပေးချင်ပါတယ်။ ။ တကယ်လို့ခင်ဗျားက history list item တွေထဲက content တွေကိုလုံးဝသေချာသိနေတာမျိုးမဟုတ်ဘူးဆိုရင်ပေါ့။ ။

history expansion mechanism ထဲမှာနောက်ထပ်မြောက်မြားစွာသော element တွေရှိနေပါသေးတယ်။ ။ ဒါပေမယ့် ဒီအကြောင်းအရာကအလွန်တရာနက်နဲဆန်းကြယ်နေပြီးတော့ ဆက်လုပ်ရင်ကျွန်တော်တို့ရဲ့ခေါင်းတွေ ပေါက်ကွဲထွက်သွားနိုင်ပါတယ်။ ။ bash man page ထဲက “HISTORY EXPANSION” section မှာအကုန်လုံးသော သွေးစွန်းနေတဲ့ (ထိတ်လန့်ဖွယ်) အသေးစိတ်အချက်အလက်တွေရှိပါတယ်။ ။ စိတ်ကြိုက်စူးစမ်းရှာဖွေနိုင်ပါတယ်။ ။

Table 8-6: History Expansion Commands

Key	Action
!!	နောက်ဆုံး command ကိုပြန်လုပ်ပေးပါတယ်။ ။ up arrow ကိုနှိပ်ပြီး Enter နှိပ်လိုက်တာကပိုလွယ်မယ် ထင်ပါတယ်။ ။
!number	history list item number ကို ပြန်လုပ်ပေးပါတယ်။ ။
!string	string စာသားတွေနဲ့စတဲ့နောက်ဆုံး history list item ကိုပြန်လုပ်ပေးပါတယ်။ ။
!?string	string စာသားတွေပါဝင်နေတဲ့နောက်ဆုံး history list item ကိုပြန်လုပ်ပေးပါတယ်။ ။

SCRIPT

bash ထဲက command history feature တွေမှာနောက်ထပ်အနေနဲ့ Linux distribution တွေအတော် များများမှာ script လို့ခေါ်တဲ့ program တခုပါဝင်ပြီးတော့အဲဒါက shell section ကြီးတခုလုံးကို record လုပ်ပြီး တော့ file တခုထဲမှာသွားသိမ်းထားပေးပါတယ်။ ။ command ရဲ့ အခြေခံ syntax ကတော့

script [file]

file ဆိုတာကတော့ record လုပ်ထားတာတွေကိုသိမ်းထားဖို့အသုံးပြုတဲ့ file ရဲ့နာမည်ဖြစ်ပါတယ်။ file ကိုမဖော်ပြခဲ့ရင် typescript ကိုသုံးသွားမှာဖြစ်ပါတယ်။ program ရဲ့ option တွေနဲ့ feature တွေရဲ့ list အပြည့်အစုံအတွက် script man page ကိုကြည့်လိုက်ပါ။

Final Note

ဒီ chapter မှာ အကြမ်းစားစာရိုက်ကျွမ်းကျင်ဆရာကြီးတွေ ၊ သူတို့အလုပ်ပိုတွေကို အကူအညီပေးဖို့အတွက် shell ကထည့်ထားပေးတဲ့ keyboard လှည့်ကွက်တချို့ကို ကျွန်တော်တို့အနေနဲ့ လေ့လာခဲ့ပြီးဖြစ်ပါတယ်။ အချိန်တွေကုန်လွန်သွားပြီးတော့ခင်ဗျားအနေနဲ့ command-line နဲ့ပိုပြီးအကျွမ်းတဝင်ရှိလာတဲ့အချိန်မှာ ခင်ဗျား ဒီ chapter ကိုပြန်လာပြီးတော့နောက်ထပ်ဒီလိုလှည့်ကွက်တွေကိုလာသယ်ဦးမယ်ဆိုတာကိုကျွန်တော်သံသယရှိနေမိတယ်။ အခုအတွက်ကတော့ သူတို့ကို ရွေးချယ်ချင်ရင်ရွေးချယ်လို့ရပြီးတော့အကူအညီကောင်းရနိုင်တာမျိုးအဖြစ်ပဲသဘောထားလိုက်ပါ။

9

PERMISSIONS

Unix ရိုးရာထဲက operating system တွေက MS-DOS ရိုးရာနဲ့ မတူကွဲပြားတာက သူတို့ဟာ multitasking system တွေဖြစ်ရုံမကဘဲ multiuser system တွေပါဖြစ်နေတာပါပဲ။

ဒါကဘာကိုအတိအကျဆိုလိုတာလဲ ? သူဆိုလိုတာက computer တလုံးကို လူတယောက်ထက်ပိုပြီးတော့ တပြိုင်နက်အသုံးပြုလို့ရတာကိုပြောတာပါ။ သာမန် computer တလုံးမှာ keyboard တခုနဲ့ monitor တခုပဲပါဖို့များပေမယ့်လည်း သူ့ကိုလူတယောက်ထက်ပိုပြီးအသုံးပြုလို့ရနိုင်ပါတယ်။ ဥပမာ - computer တလုံးဟာ network တခု သို့မဟုတ် internet နဲ့ချိတ်ဆက်ထားမယ်ဆိုရင် remote user တွေက ssh (secure shell) ကနေ log in လှမ်းဝင်ပြီးတော့ computer ကိုအသုံးပြုလို့ရပါတယ်။ တကယ်က remote user တွေအနေနဲ့ graphical application တွေကို execute လုပ်နိုင်ပြီးတော့ remote display တခုပေါ်မှာ graphical output ပေါ်လာတာကိုရရှိမှာဖြစ်ပါတယ်။ X Window System ကအဲ့ဒါမျိုးကိုသူ့ရဲ့ basic design ရဲ့အစိတ်အပိုင်းတခုအနေနဲ့ support လုပ်ပေးထားပါတယ်။

Linux ရဲ့ multiuser လုပ်ဆောင်နိုင်စွမ်းဆိုတာအခုလတ်တလောမှအသစ်ထပ်ပေါင်းပြီးဆန်းသစ်လိုက်တဲ့အရာတခုမဟုတ်ပါဘူး။ ဒါပေမယ့်သူက operating system ရဲ့ design ထဲမှာနက်နက်ရှိုင်းရှိုင်းထည့်ဝင်ထားတဲ့ feature တခုဖြစ်ပါတယ်။ Unix ကိုတည်ဆောက်ခဲ့တဲ့ environment ကနေလေ့လာကြည့်လိုက်မယ်ဆိုရင်ဒါကိုအပြည့်အဝနားလည်သွားမှာပါ။ လွန်ခဲ့တဲ့နှစ်ပေါင်းများစွာက ၊ computer တာတွေက တကိုယ်ရည်သုံးမဖြစ်သေးခင်က သူတို့ဟာ အရွယ်အစားကြီးမားပြီး ဈေးကြီးကာ ထိမ်းချုပ်ကန့်သတ်မှုတွေများခဲ့ပါတယ်။ သာမန်တက္ကသိုလ်က computer system တခုမှာဆိုရင် အဆောက်အဦတခုထဲမှာကြီးမားတဲ့ central computer (အခုခေတ် server ပေါ့)

ရှိနေပြီးတော့ terminal တွေကကျတော့ကျောင်းဝင်းရဲ့တဖက်(ကအဆောက်အဦတခုထဲမှာ) ရှိနေပြီးတော့သူတို့တွေ တခုစီက central computer ကိုချိတ်ဆက်ထားကြပါတယ်။ computer က user တွေအများကြီးကိုတပြိုင်နက်ထဲ support ပေးပါလိမ့်မယ်။

ဒါကိုလက်တွေ့လုပ်ဆောင်ကြည့်ဖို့အတွက်ဆိုရင် user တွေ အချင်းချင်းထံမှကာကွယ်ပေးဖို့အတွက်နည်းလမ်းတခုတော့ပေါ်ထွက်လာဖို့လိုပါတယ်။ နောက်ဆုံးမှာတော့ user တယောက်ရဲ့လုပ်ဆောင်ချက်တွေဟာ computer ကို crash ဖြစ်အောင်လုပ်ခွင့်မပြုတာ ဒါမှမဟုတ် user တယောက်ကအခြား user တယောက်ပိုင်ဆိုင်တဲ့ file တွေကိုအနှောက်အယှက်ပေးတာမျိုးလုပ်လို့မရတော့ပါဘူး။

အခု Chapter မှာကျွန်တော်တို့က system လုံခြုံရေးအပိုင်းမှာမရှိမဖြစ်လိုအပ်တဲ့အပိုင်းတွေကိုကြည့်သွားမှာဖြစ်ပြီးတော့ အောက်ပါ command တွေကိုမိတ်ဆက်ပေးသွားမှာဖြစ်ပါတယ်။

- **id** — user ရဲ့ identity ကိုဖော်ပြပေးမှာဖြစ်ပါတယ်။
- **chmod** — file တခုရဲ့ mode ကိုပြောင်းပေးမှာဖြစ်ပါတယ်။
- **umask** — default file ရဲ့ permission တွေကိုသတ်မှတ်ပေးမှာဖြစ်ပါတယ်။
- **su** — အခြား user တယောက်အနေနဲ့ shell တခုကို run ပေးမှာဖြစ်ပါတယ်။
- **sudo** — အခြား user တယောက်အနေနဲ့ command တခုကို Execute လုပ်ပေးမှာဖြစ်ပါတယ်။
- **chown** — file တခုရဲ့ owner ကိုပြောင်းပေးမှာဖြစ်ပါတယ်။
- **chgrp** — file တခုရဲ့ group ownership ကိုပြောင်းပေးမှာဖြစ်ပါတယ်။
- **passwd** — user ရဲ့ password ကိုပြောင်းပေးမှာဖြစ်ပါတယ်။

Owners, Group Members, and Everybody Else

ကျွန်တော်တို့ Chapter 4 မှာတုန်းက system ကိုစူးစမ်းလေ့လာခဲ့စဉ်တုန်းက /etc/shadow ထဲက file ကို စစ်ဆေးဖို့ကြိုးစားတဲ့အခါကျွန်တော်တို့အနေနဲ့အောက်ပါ ပြဿနာမျိုးကိုကြုံတွေ့ကောင်းကြုံတွေ့ခဲ့နိုင်ပါတယ်။

```
jok3r@lucy:~$ file /etc/shadow
/etc/shadow: regular file, no read permission
jok3r@lucy:~$ less /etc/shadow
/etc/shadow: Permission denied
jok3r@lucy:~$
```

ဒီ error message ပေါ်ရတဲ့အကြောင်းရင်းကတော့ ပုံမှန်အသုံးပြုသူ regular user တွေအနေနဲ့ကျွန်တော် တို့မှာဒီ file ကိုဖတ်ဖို့ခွင့်ပြုချက် permission မရှိလို့ပါပဲ ။ Unix security model ထဲမှာ user တယောက်က file တခု သို့မဟုတ် directory ကိုပိုင်ဆိုင်ကောင်းပိုင်ဆိုင်ပါလိမ့်မယ် ။ user တယောက်က file တခု သို့မဟုတ် directory ကိုပိုင်ဆိုင်တဲ့အခါမှာ user အနေနဲ့ အဲဒါတွေ (file တွေ directory တွေ) ကို access လုပ်ခွင့်ရရှိပါတယ် ။ အပြန် အလှန်အနေနဲ့ user တွေက file တွေ နဲ့ directory တွေကိုသူတို့ရဲ့ owner တွေက access ခွင့်ပြုထားတဲ့ user တယောက်သို့မဟုတ်တယောက်ထက်ပိုတဲ့ group တခုထဲမှာပါဝင်နိုင်သွားပါတယ် ။ group တခုထဲကို access တွေ့နောက်ထပ်တိုးပေးဖို့အတွက်ဆိုရင် owner အနေနဲ့လူအားလုံးကို access လုပ်ပိုင်ခွင့် set အချို့ကိုပေးကောင်း ပေးရနိုင်ပါတယ် ။ အဲဒါကို Unix terms မှာ world လို့ရည်ညွှန်းပြောဆိုလေ့ရှိပါတယ် ။ ခင်ဗျားရဲ့ identity နဲ့ပတ် သက်တဲ့သတင်းအချက်အလက်တွေကိုရှာဖွေဖို့အတွက်ဆိုရင် id command ကိုအသုံးပြုရပါတယ် ။

```
jok3r@lucy:~$ id
uid=500(jok3r) gid=500(jok3r) groups=500(jok3r)
```

output ကိုတချက်ကြည့်ကြစို့ ။ user account တွေကိုဖန်တီးတဲ့အခါမှာ user တွေဟာ user ID သို့မဟုတ် uid ဆို တဲ့ နံပါတ်တခု assign လုပ်ခြင်းခံရပါတယ် ။ အဲဒါက အဲဒီအချိန်မှာ ၊ လူသားတွေအတွက်အကူအညီဖြစ်စေဖို့အတွက် username ဆီကို mapped ညွှန်းပေးလိုက်ပါတယ် ။ user ဟာ primary group ID သို့မဟုတ် gid တခု assign လုပ် ခံရပြီးတော့ (user ဟာ) အခြားနောက်ထပ် group တွေမှာလည်းသူကပါဝင်ကောင်းပါဝင်နေနိုင်ပါတယ် ။ ပြီးခဲ့တဲ့ ဥပမာက Fedora system နဲ့လုပ်ပြထားပါတယ် ။ Ubuntu လိုအခြား system တွေမှာတော့ output ကကြည့်ရတာ အနည်းငယ်ကွဲလွဲနေနိုင်ပါတယ် ။

```
jok3r@lucy:~$ id
uid=1000(jok3r)                                gid=1000(jok3r)
groups=1000(jok3r),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),99
8(nordvpn)
jok3r@lucy:~$
```

ကျွန်တော်တို့မြင်တွေ့နေရတဲ့အတိုင်းပဲ uid နဲ့ gid နံပါတ်တွေကမတူကြပါဘူး ။ ဒါကဘာကြောင့်လဲဆို တော့ Ubuntu က 1000 နဲ့စ တဲ့အချိန်မှာ Fedora ကသူ့ရဲ့ regular user account တွေကိုနံပါတ် 500 ကနေစ လို့ပါ ပဲ ။ကျွန်တော်တို့အနေနဲ့ Ubuntu user တွေဟာ group တွေအများအပြားမှာပါဝင်နေတာကိုလည်းပဲတွေ့မြင်နိုင်ပါ တယ် ။ ဒါက Ubuntu ကနေသူ့ရဲ့ system device တွေနဲ့ service တွေကို privilege စီမံခန့်ခွဲတဲ့နည်းလမ်းနဲ့ သက်ဆိုင်ပါတယ် ။

ဒါဆိုအဲဒီ information တွေကဘယ်ကနေလာတာလဲ ? Linux ထဲကအခြားအရာပေါင်းမြောက်မြားစွာလိုပဲ သူကလည်း text file အချို့ဆီကနေလာတာဖြစ်ပါတယ်။ User account တွေကို /etc/passwd file ထဲမှာသတ်မှတ်ပြီးတော့ group တွေကိုကျတော့ /etc/group file ထဲမှာသတ်မှတ်ပါတယ်။ User account နဲ့ group တွေကိုဖန်တီးလိုက်တဲ့အခါမှာ အဲဒီ file တွေဟာ user ရဲ့ password ကိုကိုင်တွယ်ထားတဲ့ /etc/shadow နဲ့အတူတကွ modify လုပ်ခံရပါတယ်။ user account တခုချင်းစီအတွက် /etc/passwd file ကနေ user (login) name ၊ uid ၊ gid ၊ account ရဲ့ တကယ့်နာမည် ၊ home directory နဲ့ login shell တို့ကိုသတ်မှတ်ပေးပါတယ်။ တကယ်လို့ခင်ဗျားအနေနဲ့ /etc/passwd နဲ့ /etc/group တို့ကိုစစ်ဆေးကြည့်လိုက်မယ်ဆိုရင် regular user account တွေအပြင်အဲဒီမှာ superuser (uid 0) အတွက် account နဲ့အခြား system user တွေအစုံရဲ့ (account တွေကို) သတိထားမိမှာဖြစ်ပါတယ်။

Chapter 10 ထဲမှာကျနော်တို့ process အကြောင်းလေ့လာတဲ့အခါကျရင် အခြား user အချို့တွေဟာ တကယ်တော့ အလုပ်များနေကြတာကိုခင်ဗျားတွေ့မြင်ရပါလိမ့်မယ်။

များစွာသော Unix-like system တွေမှာ regular user တွေကို users လိုမျိုး common group တွေဆီကို assign ထည့်ပေးပေမယ့် ခေတ်သစ် Linux တွေမှာတော့ user ရဲ့နာမည်နဲ့တူတဲ့ unique (တူတာမရှိ ၊ ရှားပါး) ဖြစ်ပြီးတော့ single-member (အဖွဲ့ဝင်တယောက်ထဲသာပါဝင်တဲ့) group တခုကိုဖန်တီးပေးပါတယ်။ ဒါက သီးသန့် permission assignment လုပ်တဲ့အခါမှာလွယ်ကူစေပါတယ်။

Reading, Writing, and Executing

file တွေနဲ့ directory တွေကို access right (access လုပ်ပိုင်ခွင့်) နဲ့ပတ်သက်ပြီးတော့ read access ၊ write access နဲ့ execution access ဆိုပြီးသတ်မှတ်ထားပါတယ်။ တကယ်လို့ကျွန်တော်တို့အနေနဲ့ ls command ရဲ့ output ကိုတချက်ကြည့်လိုက်မယ်ဆိုရင် ဒါကိုဘယ်လို implement လုပ်ထားသလဲဆိုတာကိုကျွန်တော်တို့က သဲလွန်စအချို့ အနေနဲ့ရရှိနိုင်ပါတယ်။

```
jok3r@lucy:~$ > foo.txt
jok3r@lucy:~$ ls -l foo.txt
-rw-rw-r-- 1 jok3r jok3r 0 May 21 19:41 foo.txt
jok3r@lucy:~$
```

listing ရဲ့ပထမဦးဆုံးသော character ၁၀ လုံး ဟာ file attribute တွေဖြစ်ပါတယ်။ (Figure 9-1 ကိုကြည့်ပါ) အဲဒီ character တွေရဲ့အရှေ့ဆုံးကဟာက file type ဖြစ်ပါတယ်။ Table 9-1 မှာခင်ဗျားအနေနဲ့တွေ့မြင်ရဖို့များတဲ့ file type တွေကို list လုပ်ပြထားပါတယ်။ (အဲဒီမှာအခြားအမျိုးအစားတွေရှိပါသေးတယ် ၊ တွေ့ရခဲတဲ့ type တွေရောပေါ့)



- ❶ File type (see Table 9-1)
- ❷ Owner permissions (see Table 9-2)
- ❸ Group permissions (see Table 9-2)
- ❹ World permissions (see Table 9-2)

Figure 9-1: Breakdown of file attributes

file attribute တွေရဲ့ ကျန်တဲ့ character ၉ လုံး ကတော့ file mode လို့ခေါ်ပါတယ် ။ file owner ၊ file ရဲ့ group owner နဲ့ ကျန်တဲ့လူတွေအကုန်လုံးရဲ့ read, write နဲ့ execute permission တွေကိုကိုယ်စားပြုပါတယ် ။

(permission တွေကို) ထည့်လိုက်တဲ့အခါမှာ r , w နဲ့ x mode attribute တွေက file တွေနဲ့ directory တွေအပေါ်မှာ Table 9-2 မှာပြထားသလို တိကျတဲ့ effect တွေသက်ရောက်စေပါတယ် ။

Table 9-1: File Types

Attribute	File Type
-	ပုံမှန် regular file တခုဖြစ်ပါတယ် ။
d	directory တခုဖြစ်ပါတယ် ။
l	symbolic link တခုဖြစ်ပါတယ် ။ symbolic link နဲ့ဆိုရင်တခုသတိထားရမှာက ကျန်တဲ့ file attribute တွေက အမြဲတမ်း rwxrwxrwx ဖြစ်နေပြီးတော့အဲဒါကလည်း value အတုတွေပဲဖြစ်ပါတယ် ။ တကယ့် file attribute ရဲ့ value တွေက symbolic link ကညွှန်ပြနေတဲ့ file ထဲမှာရှိနေပါတယ် ။
c	character special file တခုဖြစ်ပါတယ် ။ ဒီ file type တွေက terminal သို့မဟုတ် modem တခုလိုမျိုး data တွေကို stream of byte တွေလိုမျိုးကိုင်တွယ်ပေးတဲ့ device တခုကိုရည်ညွှန်းပါတယ် ။
b	block special file တခုဖြစ်ပါတယ် ။ ဒီ file type တွေက hard drive သို့မဟုတ် CD-ROM drive တခုလိုမျိုး data တွေကို block တွေလိုမျိုးကိုင်တွယ်ပေးတဲ့ device တခုကိုရည်ညွှန်းပါတယ် ။

Table 9-2: Permission Attributes

Attribute	Files	Directories
r	file တခုကိုဖွင့်ပြီးဖတ်ခွင့်ပေးပါတယ် ။	တကယ်လို့ execute attribute ကိုလည်း set လုပ်ထားခဲ့မယ်ဆိုရင် directory တခုရဲ့ content တွေကိုလည်း list လုပ်ခွင့်ပြုပေးပါတယ် ။
w	file တခုကို သွားရေးတာ သို့မဟုတ် တဝက်ဖျက်ချပစ်တာမျိုးလုပ်ခွင့်ပေးပါတယ် ။ ဘယ်လိုပဲဖြစ်ဖြစ် ဒီ attribute က file တွေကို နာမည်ပြောင်းတာ ၊ ဖျက်ပစ်တာမျိုးလုပ်ခွင့်မပေးပါဘူး ။ directory attribute တွေက file တွေကို နာမည်ပြောင်းတာ ၊ ဖျက်ပစ်တာမျိုးလုပ်ခွင့်ကိုဆုံးဖြတ်ပေးပါတယ် ။	execute attribute ကိုလည်း set လုပ်ထားခဲ့မယ်ဆိုရင် directory တခုရဲ့အထဲမှာ file တွေကို တည်ဆောက်တာ ၊ ဖျက်တာ နဲ့ နာမည်ပြောင်းတာမျိုး လုပ်ခွင့်ပြုပေးပါတယ် ။
x	file တခုကို program တခုလိုသဘောထားပြီးတော့ execute လုပ်ပေးပါတယ် ။ scripting language နဲ့ရေးထားတဲ့ program file တွေကိုလည်း execute လုပ်ပေးအောင် readable	directory တခုထဲကိုဝင်ခွင့်ပြုပေးပါတယ် ။ ဥပမာ - cd directory

	အနေနဲ့မဖြစ်မနေ set လုပ်ထားရပါမယ် ။	
--	------------------------------------	--

Table 9-3 မှာ file attribute setting တွေရဲ့ ဥပမာတချို့ကိုပြထားပေးပါတယ် ။

Table 9-3: Permission Attribute Examples

File Attributes	Meaning
-rwx-----	file owner က ဖတ်တာ ၊ ရေးတာနဲ့ execute လုပ်လို့ရတဲ့ ပုံမှန် regular file တခုဖြစ်ပါတယ် ။ ကျန်တဲ့ လူအကုန်လုံးက access လုပ်ခွင့်မရှိပါဘူး ။
-rw-----	file owner က ဖတ်တာ နဲ့ ရေးတာလုပ်လို့ရတဲ့ ပုံမှန် regular file တခုဖြစ်ပါတယ် ။ ကျန်တဲ့လူအကုန်လုံးက access လုပ်ခွင့်မရှိပါဘူး ။
-rw-r--r--	file owner က ဖတ်တာနဲ့ရေးတာလုပ်လို့ရတဲ့ ပုံမှန် regular file တခုဖြစ်ပါတယ် ။ file owner group ရဲ့ member တွေက file ကိုဖတ်ခွင့်ရှိပါတယ် ။ file က အကုန်လုံးဖတ်ခွင့်ရှိတဲ့ world readable ဖြစ်ပါတယ် ။
-rwxr-xr-x	file owner က ဖတ်တာ ၊ ရေးတာ နဲ့ execute လုပ်လို့ရတဲ့ ပုံမှန် regular file တခုဖြစ်ပါတယ် ။ ဒီ file ကိုကျန်တဲ့လူအကုန်လုံးက ဖတ်တာနဲ့ execute လုပ်တာလုပ်လို့ရပါတယ် ။
-rw-rw----	file owner နဲ့ file owner group ရဲ့ member တွေသီးသန့်သာ ဖတ်တာနဲ့ရေးတာလုပ်ခွင့်ရှိတဲ့ ပုံမှန် regular file တခုဖြစ်ပါတယ် ။
Lrwxrwxrwx	symbolic link တခုဖြစ်ပါတယ် ။ symbolic link အကုန်လုံးမှာ permission အတူတွေပါပါတယ် ။ permission အစစ်တွေက symbolic link ကနေညွှန်ပြချိတ်ဆက်ထားတဲ့ file ထဲမှာထားထားပါတယ် ။
drwxrwx---	directory တခုဖြစ်ပါတယ် ။ file owner နဲ့ file owner group ရဲ့ member တွေကဒီ directory ထဲကို ဝင်ပြီးတော့ အဲ့ဒီ directory အတွင်းမှာရှိတဲ့ file တွေကို အသစ်ဆောက်တာ ၊ နာမည်ပြောင်းတာနဲ့ ဖျက်ပစ်တာတွေကိုလုပ်နိုင်ပါတယ် ။
drwxr-x---	directory တခုဖြစ်ပါတယ် ။ file owner က directory ထဲကိုဝင်ပြီးတော့ directory အတွင်းမှာရှိတဲ့ file တွေကို အသစ်ဆောက်တာ ၊ နာမည်ပြောင်းတာနဲ့ ဖျက်ပစ်တာတွေကိုလုပ်နိုင်ပါတယ် ။ file owner group ရဲ့ member တွေကတော့ directory ထဲကိုဝင်ခွင့်ရှိပေမယ့် file တွေကို အသစ်ဆောက်တာ ၊ နာမည်ပြောင်းတာနဲ့ ဖျက်ပစ်တာတွေကိုတော့လုပ်ခွင့်မရှိပါဘူး ။

chmod—Change File Mode

file သို့မဟုတ် directory တခုရဲ့ mode (permission တွေ) ကိုပြောင်းချင်တဲ့အခါမှာ chmod command ကိုအသုံးပြုပါတယ် ။ သတိထားရမှာက file ရဲ့ owner သို့မဟုတ် superuser ကသာ file သို့မဟုတ် directory တခုရဲ့ mode (permission တွေ) ကိုပြောင်းခွင့်ရှိပါတယ် ။ chmod ကမတူညီတဲ့နည်းလမ်း ၂ ခုနဲ့ mode ပြောင်းလဲသတ်မှတ်ခြင်းကို support ပေးပါတယ် ။ octal number representation နဲ့ symbolic representation တို့ဖြစ်ပါတယ် ။ ကျွန်တော်တို့ octal number representation အကြောင်းကိုအရင်စလေ့လာကြပါမယ် ။

octal number representation

octal notation နဲ့ဆိုရင် ကျွန်တော်တို့က octal number တွေကိုအသုံးပြုပြီးတော့ ကိုယ်ဖြစ်ချင်တဲ့ permission ရဲ့ pattern ကို set လုပ်ပါတယ် ။ octal number ထဲက digit တလုံးစီက binary digit ၃ လုံးကို ကိုယ်စားပြုတာကြောင့် ဒါက file mode ကိုသိမ်းထားဖို့သုံးတဲ့ scheme ဆီကိုကောင်းကောင်းလမ်းညွှန်ပေးနိုင်ပါတယ် ။ Table 9-4 ကကျွန်တော်တို့ဘာကိုဆိုလိုတာလဲဆိုတာကိုပြပေးနေပါတယ် ။

Table 9-4: File Modes in Binary and Octal

Octal	Binary	File Mode
0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	rw-
7	111	rwX

WHAT THE HECK IS OCTAL?

Octal (base 8) နဲ့သူ့ညီတဝမ်းကွဲ hexadecimal (base 16) တို့ဟာ computer တွေပေါ်မှာ နံပါတ်တွေကို ဖော်ပြတဲ့နေရာမှာမကြာခဏအသုံးပြုတဲ့ number system တွေဖြစ်ကြပါတယ် ။ ကျွန်တော်တို့လူသားတွေ ၊ ကျွန်တော်တို့ တွေ(ဒါမှမဟုတ်ကျွန်တော်တို့ထဲကအများစု)ဟာလက်ချောင်း ၁၀ ချောင်းနဲ့မွေးဖွားလာတာဖြစ်တဲ့ အတွက် ရေတွက်တဲ့အခါမှာ base 10 numbering system ကိုအသုံးပြုကြတယ်ဆိုတဲ့အမှန်တရားအပေါ် အကြွေးတင်ရှိနေပါတယ် ။ အခြားတဖက်မှာတော့ Computer တွေဟာ လက်ချောင်းတချောင်းထဲနဲ့မွေးဖွားလာတာ ဖြစ်တဲ့အတွက်ကြောင့်သူရဲ့ ရေတွက်မှုအားလုံးကို binary (base 2) ကိုအသုံးပြုပါတယ် ။ သူတို့ရဲ့ number system မှာ zero နဲ့ one ဆိုတဲ့ဂဏန်း ၂ လုံးပဲပါဝင်ပါတယ် ။ ဒါကြောင့် binary မှာရေတွက်မှုတွေကဒီလိုပုံစံဖြစ်နေကြပါတယ် ။ 0, 1, 10, 11, 100, 101, 110, 111, 1000, 1001, 1010, 1011 . . .

octal မှာဆိုရင်တော့ ရေတွက်မှုကို zero ကနေ seven အထိအသုံးပြုပါတယ် ။ ဒီလိုမျိုးပေါ့ ...

0, 1, 2, 3, 4, 5, 6, 7, 10, 11, 12, 13, 14, 15, 16, 17, 20, 21 . . .

Hexadecimal မှာကျတော့ရေတွက်မှုကို zero ကနေ nine အထိအပြင် letter A ကနေ F အထိပါထည့် သွင်းအသုံးပြုပါတယ် ။ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 10, 11, 12, 13 . . .

ကျွန်တော်တို့အနေနဲ့ binary ထဲမှာအမှန်အတိုင်းသိမြင်တဲ့အခါ (Computer မှာလည်းလက်ချောင်းတချောင်းပဲပါတဲ့အတွက်) octal နဲ့ hexadecimal တွေကဘာအတွက်ကောင်းလို့လဲ ? အဖြေကတော့လူသားတွေအတွက်အဆင်ပြေတာနဲ့သက်ဆိုင်ပါတယ် ။ အကြိမ်မြောက်မြားစွာ ၊ data အပိုင်းအစအသေးလေးတွေက Computer ပေါ်မှာ bit pattern တွေအနေနဲ့ကိုယ်စားပြုပေးပါတယ် ။ RGB color တခုကိုဥပမာအနေနဲ့ထားပါမယ် ။ Computer display အများစုပေါ်မှာ pixel တခုချင်းဆီမှာ အစိတ်အပိုင်း ၃ ခုစီပါဝင်နေပါတယ် ။ 8 bits of red, 8 bits of green နဲ့ 8 bits of blue တို့ဖြစ်ပါတယ် ။ ချစ်စရာကောင်းတဲ့ medium blue အတွက်ဆိုရင် 24-digit number ရှိနေမှာဖြစ်ပါတယ် ။ 010000110110111111001101

အဲ့ဒီလိုနံပါတ်တွေကိုရေးလိုက်ဖတ်လိုက်တနေကုန်လုပ်နေရမယ်ဆိုရင်ခင်ဗျားဘယ်လိုနေမလဲ ? ကျွန်တော်တော့မထင်ပါဘူး ။ အဲ့ဒီမှာအခြား number system ကအကူအညီဖြစ်စေတာပဲ ။ hexadecimal number တလုံးက binary number ၄ လုံးကို ကိုယ်စားပြုပေးပါတယ် ။ octal ထဲမှာဆိုရင် သူ့ဂဏန်းတလုံးက binary number ၃ လုံးကို ကိုယ်စားပြုပေးပါတယ် ။ ဒါကြောင့်ကျွန်တော်တို့ရဲ့ 24-digit medium blue ဟာ 6-digit hexadecimal number တခုအဖြစ်ကျုံ့သွားစေပါတယ် ။ 436FCD ဆိုပြီးတော့ပေါ့ ။ hexadecimal ထဲမှာသူ့ဂဏန်းတွေကိုသေချာတန်းစီပြီးထည့်ထားတာဖြစ်တဲ့အတွက် ကျွန်တော်တို့အနေနဲ့ red အတွက် အစိတ်အပိုင်း color က 43 ၊ green အတွက်က 6F နဲ့ blue အတွက်က CD ဆိုပြီးတော့မြင်နိုင်ပါတယ် ။

အခုခေတ်မှာတော့ hexadecimal notation (တခါတရံ hex လို့ခေါ်တယ်) တွေဟာ octal တွေထက်ပိုအသုံးများပါတယ် ။ ဒါပေမယ့် မကြာခင်မှာ octal ရဲ့ three bits of binary ကိုဖော်ပြနိုင်စွမ်းဟာအရမ်းအသုံးဝင်ကြောင်းကျွန်တော်တို့တွေ့မြင်ရမှာဖြစ်ပါတယ် ။

octal ဂဏန်း ၃ လုံးကိုအသုံးပြုခြင်းအားဖြင့် ကျွန်တော်တို့အနေနဲ့ owner အတွက် file mode ကို set လုပ်လို့ရပါတယ် ။

```
jok3r@lucy:~$ > foo.txt
jok3r@lucy:~$ ls -l foo.txt
-rw-rw-r-- 1 jok3r jok3r 0 May 22 15:35 foo.txt
jok3r@lucy:~$ chmod 600 foo.txt
jok3r@lucy:~$ ls -l foo.txt
-rw----- 1 jok3r jok3r 0 May 22 15:35 foo.txt
jok3r@lucy:~$
```

argument 600 ကို passing ပေးလိုက်ခြင်းအားဖြင့် ကျွန်တော်တို့က (file ကို) ဖတ်တာနဲ့ရေးတာလုပ်ဖို့ အတွက် owner ရဲ့ permission ကို set လုပ်တဲ့အချိန်မှာ group owner နဲ့ world ဆီကနေ permission တွေ အကုန်လုံးကိုပယ်ဖျက်ရုပ်သိမ်းလိုက်ပါတယ် ။ octal-to-binary mapping ကိုမှတ်မိနေဖို့ကအခက်အခဲရှိနေတာ ကြောင့် ခင်ဗျားအနေနဲ့ ထုံးစံအတိုင်းအသုံးများတဲ့ဟာတခုကိုသုံးရပါလိမ့်မယ် ။

7 (rwx), 6 (rw-), 5 (r-x), 4 (r--), and 0 (---)

Symbolic Representation

chmod က file mode တွေကိုအတိအကျသတ်မှတ်ဖို့အတွက် symbolic notation တခုကိုလည်း support လုပ်ပေးပါတယ် ။ symbolic notation ကိုအပိုင်း ၃ ပိုင်းခွဲခြားထားပါတယ် ။ change လုပ်လိုက်တာဟာဘယ်သူတွေ အပေါ်ကို effect သက်ရောက်မှုရှိမလဲ ၊ ဘာ operation တွေလုပ်ဆောင်ပေးသွားမလဲဆိုတာနဲ့ ဘယ် permission တွေ set ဖြစ်သွားမလဲဆိုတာတွေပဲဖြစ်ပါတယ် ။ ဘယ်သူတွေကို effect သက်ရောက်မှုရှိမလဲဆိုတာကိုအတိအကျ ပြောဖို့အတွက် u, g, o နဲ့ a ဆိုတဲ့ character တွေကိုပူးတွဲပြီးတော့ Table 9-5 မှာပြထားတဲ့အတိုင်းအသုံးပြုရမှာဖြစ် ပါတယ် ။

Table 9-5: chmod Symbolic Notation

Symbol	Meaning
u	user အတွက်အတိုကောက်ဖြစ်ပါတယ် ဒါပေမယ့် file သို့မဟုတ် directory ရဲ့ owner ကိုဆိုလိုပါတယ်။
g	Group owner ကိုဆိုလိုပါတယ် ။
o	others အတွက်အတိုကောက်ဖြစ်ပါတယ် ဒါပေမယ့် world ကိုဆိုလိုပါတယ်။
a	all အတွက်အတိုကောက်ဖြစ်ပါတယ် u ,g နဲ့ o ကိုပေါင်းထားတာဖြစ်ပါတယ် ။

တကယ်လို့ ဘာ character မှမပြထားရင် all လို့မှတ်ယူလိုက်မှာပါ ။ operation တကယ်လုပ်ဆောင်ချိန် မှာတော့ a+ ဆိုရင် permission တခုထပ်ပေါင်းထည့်မှာဖြစ်ပြီး a- ဆိုရင် permission တခုထုတ်ပယ်မှာဖြစ်ပါတယ် ။ ဒါမှမဟုတ် a= ဆိုရင်တော့သတ်မှတ်ထားတဲ့ permission တွေသာသက်ရောက်စေမှာဖြစ်ပြီးတော့ကျန်တာ အကုန်လုံးကိုထုတ်ပယ်လိုက်မှာဖြစ်ပါတယ် ။

permission တွေကို r , w နဲ့ x character တွေအသုံးပြုပြီးတော့တိတိကျကျသတ်မှတ်ပါတယ် ။ Table 9-6 မှာ symbolic notation တချို့ကို list လုပ်ပြထားပါတယ် ။

Table 9-6: chmod Symbolic Notation Examples

Notation	Meaning
u+x	owner ကို execute permission ပေါင်းထည့်ပေးတာပါ ။
u-x	owner ဆီက execute permission ပယ်ထုတ်တာပါ ။
+x	owner, group နဲ့ world ကို execute permission ပေါင်းထည့်ပေးတာပါ ။ a+x နဲ့တူပါတယ် ။

o-rw	owner နဲ့ group owner ကလွဲရင်ကျန်တဲ့လူတွေအကုန်လုံးဆီက read နဲ့ write permission တွေ ပယ်ထုတ်တာပါ။
gO=rw	owner နဲ့ owner ကလွဲရင်ကျန်တဲ့လူတွေအကုန်လုံးကို read နဲ့ write permission တွေပေးတာပါ။ တကယ်လို့ group owner သို့မဟုတ် world မှာအရင်ကကြိုပေးပြီးသား execute permission တွေရှိနေရင်သူက ပယ်ထုတ်လိုက်မှာပါ။
u+x,gO=rx	owner ကို execute permission ပေးပြီးတော့ group and others တွေအတွက် read and execute permission တွေထည့်ပေးတာပါ။ သတ်မှတ်ချက်အများကြီးထည့်မယ်ဆိုရင် comma (,) နဲ့ခြားပေးရပါမယ်။

တချို့လူတွေကတော့ octal notation ကိုပိုသုံးချင်ကြပါတယ်။ တချို့ဘဲကြီးတွေက symbolic ကိုအရမ်းကြိုက်ကြပါတယ်။ Symbolic notation ကလည်း others တွေကိုအနှောက်အယှက်မဖြစ်စေပဲ single attribute တခု set လုပ်တဲ့နေရာမှာ အားသာချက်တွေကိုရရှိစေနိုင်ပါတယ်။

chmod man page မှာ အသေးစိတ်အချက်အလက်နဲ့ option တွေရဲ့ list ကိုတချက်လောက်ကြည့်လိုက်ပါ။ --recursive option နဲ့ပက်သက်ပြီးတော့သတိပေးစကားပြောချင်တာက၊ သူက file နဲ့ directory ၂ခုစလုံးပေါ်မှာ လုပ်လုပ်တဲ့အတွက် လူတယောက်မျှော်လင့်ထားသလောက်အသုံးဝင်မှာမဟုတ်ပါဘူး။ ဘာလို့လဲဆိုတော့ ကျွန်တော်တို့အနေနဲ့ file နဲ့ directory ၂ခုစလုံးပေါ်မှာတူညီတဲ့ permission ပေးတယ်ဆိုတာ ရှားပါတယ်။

Setting File Mode with the GUI

အခုဆိုရင် ကျွန်တော်တို့အနေနဲ့ file နဲ့ directory တွေအပေါ်မှာ permission တွေဘယ်လို set လုပ်တယ်ဆိုတာကို မြင်တွေ့ခဲ့ပြီးပြီဖြစ်တဲ့အတွက် ကျွန်တော်တို့အနေနဲ့ GUI ထဲက permission dialogue ကိုသိဖို့ကောင်းနေပါပြီ။ Nautilus (GNOME) နဲ့ Konqueror(KDE) နှစ်ခုစလုံးထဲမှာ file သို့မဟုတ် directory ပေါ်မှာ right-click နှိပ်ခြင်းဖြင့် properties dialogue ထွက်ပေါ်လာမှာ ဖြစ်ပါတယ်။

ဒီမှာကျွန်တော်တို့က owner, group နဲ့ world တို့အတွက် setting တွေကို တွေ့မြင်နိုင်ပါတယ်။ KDE ထဲမှာဆိုရင်တော့ Advanced Permissions button ကို နှိပ်လိုက်ရင် mode attribute တွေကိုတခုချင်းစီ set လုပ်ပေးနိုင်တဲ့နောက်ထပ် dialogue ထွက်ပေါ်လာမှာဖြစ်ပါတယ်။ command line ကိုနားလည်ခြင်းကြောင့်ရရှိလာတဲ့နောက်ထပ်အောင်ပွဲတခုပဲဖြစ်ပါတယ်။

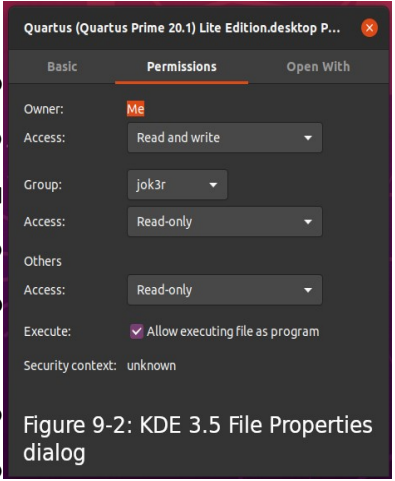


Figure 9-2: KDE 3.5 File Properties dialog

umask—Set Default Permissions

file တခုကို တည်ဆောက်လိုက်တဲ့အချိန်မှာပါလာတဲ့ default permission ကို umask command က control လုပ်ပါတယ်။ သူက octal notation တွေကိုအသုံးပြုပြီးတော့ file တခုရဲ့ mode attribute တွေကို ပယ်ဖျက်ရာမှာ mask ရဲ့ bit တွေကိုဖော်ပြပေးပါတယ်။

```
jok3r@lucy:~$ rm -f foo.txt
jok3r@lucy:~$ umask
0002
jok3r@lucy:~$ > foo.txt
jok3r@lucy:~$ ls -l foo.txt
-rw-rw-r-- 1 jok3r jok3r 0 May 22 17:39 foo.txt
jok3r@lucy:~$
```

ကျွန်တော်တို့ပထမအနေနဲ့ အသစ်ပြန်စတာသေချာအောင် ရှိပြီးသား foo.txt ကိုဖျက်ပစ်ပါမယ် ။ နောက်ထပ် ၊ ကျွန်တော်တို့ current value ကိုသိဖို့အတွက် umask command ကို argument မပါပဲနဲ့ ran ကြည့်ပါမယ် ။ သူက value 0002 (value 0022 ဆိုတာကလည်းနောက်ထပ် common default value ပဲဖြစ်ပါတယ်)နဲ့ respond ပြန်လာပြီး အဲဒါကလည်းကျွန်တော်တို့ mask ရဲ့ octal representation ပဲဖြစ်ပါတယ် ။ အဲဒါမှာ ကျွန်တော်တို့က foo.txt file တခုကိုအသစ်တည်ဆောက်လိုက်ပြီးတော့ သူ့ရဲ့ permission တွေကိုစစ်ဆေးကြည့်ပါမယ် ။

ကျွန်တော်တို့အနေနဲ့ အခြားသူတွေအကုန်လုံးက read permission တခုထဲရနေတဲ့အချိန်မှာ owner နဲ့ group ကတော့ read and write permission တွေရနေတာကိုတွေ့မြင်ရမှာပါ ။ world မှာ write permission မရတာက mask ရဲ့ value ကြောင့်ဖြစ်ပါတယ် ။ ကျွန်တော်တို့ရဲ့ဥပမာကိုနောက်တကြိမ်ထပ်လုပ်ပါမယ် ။ အခုတကြိမ်မှာတော့ ကျွန်တော်တို့ mask ရဲ့ setting ကိုကျွန်တော်တို့ကိုယ်တိုင်လုပ်မှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ rm foo.txt
jok3r@lucy:~$ umask 0000
jok3r@lucy:~$ > foo.txt
jok3r@lucy:~$ ls -l foo.txt
-rw-rw-rw- 1 jok3r jok3r 0 May 22 17:51 foo.txt
jok3r@lucy:~$
```

ကျွန်တော်တို့က mask ကို 0000 (သူ့ကိုထိထိရောက်ရောက်ပိတ်ချလိုက်တာ) မှာ set ထားလိုက်တဲ့အခါ ကျွန်တော်တို့အနေနဲ့ file ဟာအခုအချိန်မှာ world က write လုပ်လို့ရသွားတာကိုတွေ့မြင်နိုင်ပါတယ် ။ ဒါဘယ်လို အလုပ်လုပ်သလဲဆိုတာကိုနားလည်ဖို့အတွက်ဆိုရင် ကျွန်တော်တို့အနေနဲ့ octal number တွေကိုတချက်ပြန်ကြည့်ရပါလိမ့်မယ် ။ တကယ်လို့ကျွန်တော်တို့က mask ကို binary အဖြစ် expand လုပ်ချပြီးတော့ သူ့ရဲ့ attribute တွေနဲ့ နှိုင်းယှဉ်ကြည့်မယ်ဆိုရင် ဘာတွေဖြစ်နေတာလဲဆိုတာကိုကျွန်တော်တို့မြင်တွေ့နိုင်မှာဖြစ်ပါတယ် ။

Original file mode	--- rw- rw- rw-
Mask	000 000 000 010
Result	--- rw- rw- r--

အရှေ့ဆုံးကသည့် 0 တွေကိုခဏမေ့ထားလိုက်ပါ။ (တမိနစ်လောက်နေရင်အဲ့ဒါကိုလေ့လာပါမယ်) ပြီးတော့ ကျွန်တော်တို့ရဲ့ mask ထဲမှာ 1 ရှိနေတဲ့နေရာကိုသေချာကြည့်လိုက်ရင် attribute တခုပယ်ထုတ်ခြင်းခံထားရပါတယ်။ အခုကိစ္စမှာဆိုရင်တော့ world ရဲ့ write permission ကိုပေါ့။ mask လုပ်တဲ့အလုပ်ကအဲ့ဒါပါပဲ။ mask ရဲ့ binary value တွေထဲမှာ 1 တလုံးပေါ်လာတိုင်း attribute တခုပယ်ထုတ်ခြင်းခံရပါတယ်။ တကယ်လို့ ကျွန်တော်တို့က mask ရဲ့ value 0022 ကိုကြည့်လိုက်မယ်ဆိုရင် သူ့ဘာလုပ်သလဲဆိုတာကို ကျွန်တော်တို့မြင်နိုင်ပါတယ်။

Original file mode	--- rw- rw- rw-
Mask	000 000 000 010
Result	--- rw- rw- r--

အခုတခါမှာလည်း binary value ထဲမှာ 1 တလုံးပေါ်လာတဲ့နေရာတိုင်းမှာ သက်ဆိုင်ရာ attribute ဟာပယ်ဖျက်ခံရပါတယ်။ value တချို့ (7 တွေထည့်ကြည့်ပါ) နဲ့ကစားကြည့်ပြီးသူ့ဘယ်လိုအလုပ်လုပ်တယ်ဆိုတာကိုအကျင့်ရသွား (သိသွား)အောင်လုပ်ကြည့်ပါ။ ခင်ဗျားကစားလို့ပြီးသွားတဲ့အခါ ပြန်ရှင်းခဲ့ဖို့မမေ့ပါနဲ့။

jok3r@lucy:~\$ umask 0002

အချိန်တော်တော်များများမှာတော့ခင်ဗျားအနေနဲ့ mask ကိုပြောင်းစရာမလိုပါဘူး။ distribution မှာပါလာပြီးသားဖြစ်တဲ့ default (mask) ကအဆင်ပြေပါတယ်။ တချို့ high-security အခြေအနေတွေမှာဆိုရင်တော့ ဘယ်လိုပဲဖြစ်ဖြစ်ခင်ဗျားအနေနဲ့ အဲ့ဒါကို ထိန်းချုပ်ကိုင်တွယ်ချင်မှာပါ။

SOME SPECIAL PERMISSIONS

ကျွန်တော်တို့အနေနဲ့ပုံမှန်အားဖြင့် octal permission mask တခုကို ဂဏန်း ၃ လုံး three-digit number နဲ့ဖော်ပြကြပေမယ့် ဂဏန်း ၄ လုံး four digits နဲ့ဖော်ပြတာက နည်းပညာသဘောအရပိုပြီးမှန်ကန်ပါတယ်။ ဘာဖြစ်လို့လဲ ? ဘာလို့လဲဆိုတော့ သူ့မှာ read, write နဲ့ execute permission တွေအပြင်အဲ့ဒီမှာအသုံးနည်းတဲ့ permission setting တွေရှိနေသေးလို့ပါပဲ။

အဲဒါတွေထဲကပထမဆုံးတခုကတော့ setuid bit (octal 4000) ဖြစ်ပါတယ် ။ executable file တခုမှာအဲဒီ permission ကို apply လုပ်လိုက်တဲ့အခါ သူက program ကိုပိုင်ဆိုင်တဲ့ owner ရဲ့ real user (program ကိုတကယ် အသုံးပြုနေသူ) ရဲ့ effective user ID ကို set လုပ်ပေးပါတယ် ။ အသုံးများတာကတော့ သူ့ကို superuser (admin) ကပိုင်ဆိုင်တဲ့ program အနည်းငယ်မှာ ပေးထားပါတယ် ။ ရိုးရိုးသာမန် user တယောက်က program တခုကို setuid root နဲ့ run လိုက်တဲ့အခါမှာ program က superuser (admin) ရဲ့ privileges လုပ်ပိုင်ခွင့်အနေနဲ့ run ပေး သွားပါတယ် ။ ဒါက program ကို ရိုးရိုးသာမန် user တယောက်အနေနဲ့ဆိုရင်တားမြစ်ပိတ်ပင်ခြင်းခံထားရတဲ့ file တွေနဲ့ directory တွေကို access လုပ်ခွင့်ပေးလိုက်တာဖြစ်ပါတယ် ။ ရှင်းပါတယ် ၊ ဘာလို့လဲဆိုတော့ အဲဒါက လုံခြုံရေးနဲ့ပတ်သက်တာတွေများလာလို့ပါပဲ ။ setuid program အရေအတွက်ဟာ အနည်းဆုံးဖြစ်နေအောင် ကိုင်တွယ်ထားဖို့လိုအပ်ပါတယ် ။

ဒုတိယအသုံးအနည်းဆုံး setting ကတော့ setgid bit (octal 2000) ဖြစ်ပါတယ် ။ ဒါက setuid bit လိုပဲ ၊ file ကိုပိုင်ဆိုင်တဲ့ owner ရဲ့ real group ID ရဲ့ effective group ID ကိုပြောင်းလဲပေးပါတယ် ။ တကယ်လို့ setgid bit ကို directory ပေါ်မှာ set လုပ်ထားပြီးတော့အဲဒီ directory ထဲမှာ file အသစ်တွေတည်ဆောက်လိုက်မယ်ဆိုရင် directory ကိုတည်ဆောက်တဲ့လူရဲ့ group ownership ထဲရောက်သွားပြီးတော့ file အသစ်တွေတည်ဆောက်တဲ့လူ က group ownership ကိုမရရှိတော့ပါဘူး ။ shared လုပ်ထားတဲ့ directory တခုထဲမှာဆိုရင် common group တခု ထဲက member တွေက directory ထဲက file တွေကို access လုပ်ဖို့လိုအပ်လာတဲ့အခါမှာ ဒါကအသုံးဝင်ပါတယ် ။ file ကိုပိုင်ဆိုင်တဲ့ owner ရဲ့ primary group ကတော့ဘာပဲဖြစ်ဖြစ်မရရှိပါဘူး ။

တတိယမြောက်ကတော့ sticky bit (octal 1000) ဖြစ်ပါတယ် ။ ဒါက ရှေးဟောင်း Unix ခေတ်ကထဲက စွဲကိုင်လာတာဖြစ်ပါတယ် ။ executable file တခုကို not swappable အဖြစ် mark လုပ်လို့ရပါတယ် ။ file တွေပေါ် မှာဆိုရင်တော့ Linux က sticky bit တွေကို မသိကျိုးကျွန်ပြုထားပါတယ် ။ directory တွေပေါ်မှာ apply လုပ်မယ် ဆိုရင်တော့ သူက user တွေက file တွေကို နာမည်ပြောင်းတာ ၊ ဖျက်တာတွေလုပ်ခြင်းကနေတားဆီးပေးပါတယ် ။ တကယ်လို့ user က directory owner ၊ file owner သို့မဟုတ် superuser မဟုတ်ဘူးဆိုရင်ပေါ့ ။ ဒါကို /tmp လို share လုပ်ထားတဲ့ directory တခုကို ထိန်းချုပ်တဲ့အခါမှာမကြာခဏအသုံးပြုပါတယ် ။

ဒီမှာ chmod ကို symbolic notation နဲ့တွဲပြီးတော့ special permission တွေ set လုပ်တဲ့အခါမှာသုံးတဲ့ ဥပမာအချို့ကိုပြထားပေးပါတယ် ။ ပထမအနေနဲ့ program တခုကို setuid သွားပြီး assign လုပ်ပါမယ် ။

chmod u+s program

ပြီးရင် directory တခုကို setgid သွားပြီး assign လုပ်ပါမယ် ။

chmod g+s dir

နောက်ဆုံးမှာ directory တခုကို sticky bit သွားပြီး assign လုပ်ပါမယ် ။

chmod +t dir

Is ကပြန်ထုတ်ပေးတဲ့ output တွေကိုကြည့်ပြီးတော့ခင်ဗျားအနေနဲ့ special permission တွေကိုဆုံးဖြတ် လို့ရပါတယ်။ ဒီမှာဥပမာအချို့ပြထားပေးပါတယ်။ ပထမအနေနဲ့ setuid (အောက်ကအတိုင်းရှိတဲ့) program တခု။

-rwsr -xr -x

အခု setgid attribute ရှိနေတဲ့ directory တခု။

drwxrwsr -x

နောက်ဆုံးမှာ sticky bit set ရှိနေတဲ့ directory တခု။

drwxrwxrwt

Changing Identities

အခြေအနေမျိုးစုံမှာ ၊ ကျွန်တော်တို့အနေနဲ့ အခြား user တယောက်ရဲ့ identity ကိုယူဖို့လိုအပ်နိုင်ပါတယ်။ မကြာခဏဆိုသလိုပဲကျွန်တော်တို့အနေနဲ့ administrative task တွေကိုလုပ်ဆောင်ဖို့အတွက် superuser privilege တွေကိုရရှိဖို့လိုအပ်ပါတယ်။ ဒါပေမယ့် ဒါကအခြား regular user အဖြစ်နဲ့ အဲဒီလို task တွေကို account ကို test လုပ်တာမျိုး လုပ်ဆောင်လို့လည်းရပါတယ်။ အခြား identity ကိုယူဖို့နည်းလမ်း ၃ ခု ရှိပါတယ်။

- logout ထွက်ပြီး အခြား user အနေနဲ့ login ပြန်ဝင်တာ။
- su command သုံးတာ။
- sudo command သုံးတာ။

ကျွန်တော်တို့အနေနဲ့ပထမနည်းလမ်းကိုကျော်ခဲ့လိုက်ပါမယ် လာလို့လဲဆိုတော့ ဘယ်လိုလုပ်ရတယ်ဆိုတာ ကိုခင်ဗျားသိပြီးသားဖြစ်တဲ့အပြင်အဲနည်းကအခြားနည်းလမ်း ၂ ခုကိုအဆင်မပြေစေလို့ပါပဲ။ ခင်ဗျားရဲ့ကိုယ်ပိုင် shell session အတွင်းကနေ su command ကခင်ဗျားကို အခြား user တယောက်အဖြစ်မှတ်ယူပြီးတော့ အဲဒီ user ရဲ့ ID နဲ့ shell session အသစ်တခုစတင်ပေးတာ သို့မဟုတ် အဲဒီ user အနေနဲ့ single command ထုတ်ပေးတာ အဲဒီ ၂ မျိုးထဲကတခုကိုလုပ်ပေးမှာဖြစ်ပါတယ်။ sudo command ကတော့ administrator တယောက်ကို /etc/sudoers လို့ခေါ်တဲ့ configuration file တွေ setup လုပ်ပေးပြီးတော့ ရိုးရိုး user တွေဆိုရင် သတ်မှတ်ထားတဲ့ identity နဲ့မှ execute လုပ်ခွင့်ပြုတဲ့ တိကျတဲ့ command တွေကိုသတ်မှတ်ပေးပါတယ်။ ဘယ် command ကိုသုံးရမလဲဆိုတာ က ခင်ဗျားသုံးနေတဲ့ Linux distribution ပေါ်မှာအများကြီးမူတည်ပြီးဆုံးဖြတ်ပါတယ်။ ခင်ဗျားရဲ့ distribution မှာ ၂ ခုစလုံးပါနိုင်ပါတယ်။ ဒါပေမယ့် သူ့ရဲ့ configuration ကဘယ်တခုကိုပိုအသားပေးသလဲဆိုတာရှိပါတယ်။ ကျွန်တော်တို့ su command နဲ့ လိုက်ပါမယ်။

su—Run a Shell with Substitute User and Group IDs

su command က shell ကိုအခြား user တယောက်အနေနဲ့စတင်ပေးပါတယ် ။ command syntax ပုံစံ ကတော့ဒီလိုပုံစံမျိုးပါ ။

su [-[l]] [user]

တကယ်လို့ -l option ထည့်လိုက်မယ်ဆိုရင် ထွက်လာမယ့် shell session သည် သတ်မှတ်ထားတဲ့ user ရဲ့ login shell တခုဖြစ်နေပါမယ် ။ ဆိုလိုတာက user ရဲ့ environment တက်လာပြီးတော့ working directory က user ရဲ့ home directory ကိုပြောင်းသွားမှာဖြစ်ပါတယ် ။ user ကိုမသတ်မှတ်ပေးရင်တော့ superuser အဖြစ် မှတ်ယူလိုက်မှာပါ ။ တခုမှတ်ထားရမှာက (ထူးဆန်းစွာပေါ့) -l ဟာ - ရဲ့အတိုကောက်ဖြစ်ပြီးတော့အဲဒါကလည်း မကြာခဏအသုံးများပါတယ် ။ superuser အတွက် shell တခုကိုစတင်ဖို့ဆိုရင်ကျွန်တော်တို့ကဒီလိုလုပ်ပါမယ် ။

(Ubuntu 20.04 မှာတော့ su အနေနဲ့ run ချင်ရင် sudo -s လို့ရိုက်ရပြီးတော့ su - အနေနဲ့ ran ချင်ရင်တော့ sudo -i လို့ရိုက်ရပါမယ် ။ ကျွန်တော့်စက်နဲ့စမ်းထားတာပဲထည့်ပေးမှာဖြစ်လို့ မူရင်းစာအုပ်နဲ့ကွဲလွဲ နေမှာဖြစ်ပါတယ် ။ အလုပ်လုပ်ပုံသဘောတရားကတော့အတူတူပါပဲ ။)

```
jok3r@lucy:~$ sudo -s
[sudo] password for jok3r:
root@lucy:/home/jok3r# exit
exit
jok3r@lucy:~$ sudo -i
root@lucy:~# exit
logout
jok3r@lucy:~$
```

command ကိုရိုက်ထည့်လိုက်ရင်ကျွန်တော်တို့ကို superuser ရဲ့ password တောင်းတဲ့ prompt တက် လာတာကိုမြင်ရမှာပါ ။ တကယ်လို့အဲဒါကိုမှန်အောင်ထည့်ပေးနိုင်ခဲ့ရင် shell prompt အသစ်တခုပေါ်လာပြီးအဲဒီ shell မှာ superuser privilege တွေ (\$ အစား # ကိုပြနေမယ်) ရှိနေကြောင်းပြနေပြီးတော့ current working directory ဟာလည်းအခုဆိုရင် superuser ရဲ့ home directory (ပုံမှန်အားဖြင့်တော့ /root) မှာရောက်နေပါလိမ့် မယ် ။ shell အသစ်ထဲကိုရောက်တာနဲ့တပြိုင်နက်ကျွန်တော်တို့အနေနဲ့ command တွေကို superuser အနေနဲ့ဆက် သုံးသွားလို့ရပါပြီ ။ ပြီးသွားရင်တော့ နကိုမူလ shell ဆီကိုပြန်သွားဖို့အတွက် exit လို့ရိုက်ပေးရပါတယ် ။

(single command တခုကို execute လုပ်ဖို့အတွက် နောက်ထပ် command တွေတကြောင်းစီမှာလိုက် ရိုက်ထည့်နေမယ့်အစား sudo command ကိုအခုလိုသုံးလို့ရပါတယ်။

sudo bash -c 'command' (သို့မဟုတ်) **sudo sh -c 'command'**

ဒီလိုပုံစံကိုသုံးလိုက်ခြင်းအားဖြင့် single command line တစ်ခုကို shell အသစ်တစ်ခုဆီမှာ execution လုပ်ဖို့ အတွက် pass ပေးလိုက်ပါတယ် ။ ကျွန်တော်တို့က ကျွန်တော်တို့ရဲ့ expansion ကို shell မှာမလုပ်စေပဲ shell အသစ်နေရာမှာသွားလုပ်စေချင်တာကြောင့် command ဆိုတဲ့နေရာကို single quote (') ၊ ခုအတွင်းမှာထားဖို့ အရေးကြီးပါတယ် ။

```
jok3r@lucy:~$ sudo bash -c 'ls -l /root/*'
[sudo] password for jok3r:
total 40
drwxr-xr-x 5 root root 4096 Jul 12  2021 arduino
drwxr-xr-x 4 root root 4096 May 23 19:36 chromium
drwxr-xr-x 5 root root 4096 Sep 18  2021 easy-openvpn-server
drwxr-xr-x 5 root root 4096 Feb  5 19:30 gimp
drwxr-xr-x 5 root root 4096 Nov 19  2021 gnome-system-monitor
drwxr-xr-x 3 root root 4096 Mar 15 20:59 sickgear
drwxr-xr-x 5 root root 4096 Nov 23  2021 snap-store
drwxr-xr-x 5 root root 4096 Apr 26 21:15 telegram-desktop
drwxr-xr-x 3 root root 4096 Mar 15 20:59 tor-middle-relay
drwxr-xr-x 5 root root 4096 Sep 19  2021 ufw
jok3r@lucy:~$
```

)

sudo—Execute a Command as Another User

sudo command က su command နဲ့နေရာအတော်များများမှာဆင်တူပေမယ့်သူ့မှာအရေးကြီးတဲ့အပို လုပ်ဆောင်နိုင်စွမ်းတချို့ရှိပါတယ် ။ administrator အနေနဲ့ sudo ကို configure လုပ်ပြီးတော့ ရိုးရိုးသာမန် user ကို အခြား user တယောက် (ပုံမှန်အားဖြင့်တော့ superuser)အနေနဲ့ အလွန်ထိမ်းချုပ်ထားတဲ့နည်းလမ်းနဲ့ command တွေကို execute လုပ်ခွင့်ပေးပါတယ် ။ အတိအကျဆိုရင် user တယောက်ကိုသတ်မှတ်ထားတဲ့ command တခု သို့မဟုတ် တခုထက်ပိုတဲ့ဆီကနေ ကန့်သတ်ခံထားရပါတယ် ။ နောက်ထပ်အရေးကြီးတဲ့ ခြားနားချက်ကတော့ sudo ကိုသုံးတဲ့အခါ superuser ရဲ့ password ကို access လုပ်ဖို့မလိုဘူး ။ sudo ကိုသုံးပြီး တော့ authenticate လုပ်ဖို့အတွက် user ကသူ့ရဲ့ကိုယ်ပိုင် password ကိုပဲထည့်ပေးရမှာဖြစ်ပါတယ် ။ ဆိုကြပါစို့ ၊ ဥပမာ sudo က superuser privilege တွေလိုအပ်တဲ့ backup_script လို့ခေါ်တဲ့ backup program အတုတခုကို run ဖို့အတွက် configure လုပ်ထားပြီး ကျွန်တော်တို့ကိုခွင့်ပြုထားတယ်ပေါ့ ။

sudo နဲ့ဆိုရင်အဲ့ဒါကိုဒီလိုလုပ်ရမှာပါ။

(ဒီနေရာမှာ backup_script ဆိုတဲ့ program ကစက်ထဲမှာတကယ်ရှိမနေပါဘူး။ ဒါကြောင့် run လိုက်ရင် sudo: backup_script: command not found ဆိုပြီးပြနေမှာဆိုပေမယ့် ခင်ဗျားလုပ်တာမှန်ပါတယ်။ ဒါကစမ်းသပ်ကြည့်တဲ့သဘောပဲ။ တကယ်လို့အဲ့ဒီ program ကရှိနေခဲ့မယ်ဆိုရင် System Backup Starting... ဘာညာဆိုပြီး run သွားမှာဖြစ်ပါတယ်။ အခု ဥပမာမှာတော့ကျွန်တော်က မ run တာနဲ့ပဲပြထားပါတယ်။)

```
jok3r@lucy:~$ sudo backup_script
[sudo] password for jok3r:
sudo: backup_script: command not found
jok3r@lucy:~$
```

command ကိုရိုက်ထည့်လိုက်တဲ့အခါမှာကျွန်တော်တို့ကို password (superuser ရဲ့ဟာမဟုတ်ပါ) တောင်းတဲ့ prompt ပေါ်လာပြီးတော့ authentication လုပ်တာပြီးသွားတာနဲ့ သတ်မှတ်ထား (ရိုက်ထည့်လိုက်တဲ့) command ကဆက်လက်လုပ်ဆောင်သွားပါတယ်။ su နဲ့ sudo ကြားကအရေးကြီးတဲ့ခြားနားချက်ကတော့ sudo က shell အသစ်တစ်ခုကိုစတင်တာ ဒါမှမဟုတ် အခြား user တယောက်ရဲ့ environment ကို load ခေါ်ပေးတာမျိုးမလုပ်ပေးပါဘူး။ ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ command တွေအနေနဲ့ sudo ကိုမသုံးပဲနဲ့ဖြစ်လာမယ့်ပိုမိုတဲ့ခြားနားချက်တွေအတွက် copy ကူးယူဖို့မလိုအပ်ပါဘူးလို့ဆိုလိုတာပါပဲ။ တခုသတိထားရမှာက ဒီလိုအပြုအမူမျိုးတွေက option မျိုးစုံကိအတိအကျထည့်ခြင်းအားဖြင့် overwrite ဖြစ်သွားနိုင်ပါတယ်။ sudo man page မှာ အသေးစိတ်သွားဖတ်ပါ။ sudo ကနောက် privilege တွေပေးထားသလဲဆိုတာကိုသိဖို့ -l option (သို့မဟုတ် --list) ကိုသုံးပြီးသူတို့ကို list ထုတ်ကြည့်ပါ။

```
jok3r@lucy:~$ sudo -l
[sudo] password for jok3r:
Matching Defaults entries for jok3r on lucy:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
```

User jok3r may run the following commands on lucy:

(ALL : ALL) ALL

```
jok3r@lucy:~$
```

UBUNTU AND SUDO

ပုံမှန် user တွေအတွက်နောက်တဖန်ပြဿနာဖြစ်လာစေတာတွေထဲကတခုကတော့ superuser privilege လိုအပ်တဲ့ သတ်မှတ်ထားတဲ့ task တွေကို ဘယ်လိုလုပ်ဆောင်ရမလဲဆိုတာပဲဖြစ်ပါတယ်။ ။ အဲဒီ task တွေထဲမှာ software တွေကို installing လုပ်တာနဲ့ updating လုပ်တာ၊ system configuration file တွေကို editing လုပ်တာ နဲ့ device တွေကို access လုပ်တာတွေလည်းပါဝင်ပါတယ်။ ။ Windows လောကမှာတော့ တခါတလေ user တွေကို administrative privilege ပေးလိုက်ခြင်းအားဖြင့်လုပ်ဆောင်ပါတယ်။ ။ ဒါက user ကို ဒီ task တွေလုပ်ဆောင်ခွင့်ပြု လိုက်တာပဲဖြစ်ပါတယ်။ ။ ဘာပဲဖြစ်ဖြစ် ဒါကလည်းပဲ user က program ကို execute လုပ်ပြီးတော့တူညီတဲ့လုပ်နိုင် စွမ်းကိုရရှိစေတာပါပဲ။ ။ ဒါက ကိစ္စတော်တော်များများမှာစိတ်ကျေနပ်စရာကောင်းပေမယ့်လည်း ဒါက virus လိုမျိုး malware (malicious software) တွေကို computer မှာစိတ်ကြိုက်လွှတ်လပ်စွာ run ခွင့်ပေးလိုက်တာလည်းဖြစ်ပါ တယ်။ ။

Unix လောကမှာတော့ အဲဒီမှာအမြဲတမ်းပဲ ရိုးရိုးသာမန် user တွေနဲ့ administrator တွေကြားမှာပိုမို ကြီးမားတဲ့ခွဲခြားမှုကြီးရှိနေပါတယ်။ ။ (အဲဒီအချက်ကြောင့်လည်း) ကျွန်တော်တို့က Unix ရဲ့ multiuser heritage ပေါ် မှာ (ကျေးဇူး) အကြွေးတင်နေပါတယ်။ ။ Unix နည်းလမ်းနဲ့ချဉ်းကပ်ဖို့ဆိုရင် superuser privilege တွေကိုလိုအပ်မှ သာပေးအပ်ရမှာဖြစ်ပါတယ်။ ။ ဒါကိုလုပ်ဖို့ဆိုရင် su နဲ့ sudo command တွေကိုအများအားဖြင့်သုံးရပါလိမ့်မယ်။ ။

လွန်ခဲ့တဲ့နှစ်တွေမတိုင်ခင်ကအထိ Linux distribution တွေကဒီကိစ္စတွေအတွက် su ကိုပိုမိုခိုခွဲကြပါတယ် ။ su က sudo လိုမျိုး configuration တွေမလိုအပ်ပါဘူး။ ။ ပြီးတော့ root account တခုပိုင်ဆိုင်ထားတာဟာလည်း Unix မှာ ရိုးရာလိုဖြစ်နေပါတယ်။ ။ ဒါက ပြဿနာတခုကိုဖြစ်ပေါ်စေပါတယ်။ ။ user တွေဟာမလိုအပ်ပဲနဲ့ root အဖြစ် လုပ်ဆောင်ဖို့ တွန်းအားပေးခံကြရပါတယ်။ ။ တကယ်တော့ တချို့ user တွေကသူတို့ရဲ့ system ကို root အနေနဲ့သီး သန့်လုပ်ဆောင်ကြတာဘာလို့လဲဆိုတော့ စိတ်ရှုပ်စရာကောင်းတဲ့ “permission denied” message တွေဆီကနေ လွတ်မြောက်နိုင်လို့ပဲဖြစ်ပါတယ်။ ။ အဲဒီလိုနည်းလမ်းနဲ့ခင်ဗျားက Linux system ရဲ့ security ကို Windows system လိုမျိုးလျှော့ချလိုက်တာပဲဖြစ်ပါတယ်။ ။ သိပ်ကောင်းတဲ့အကြံတော့မဟုတ်ပါဘူး။ ။

Ubuntu ကိုစတင်မိတ်ဆက်လာတဲ့အခါမှာ သူ့ရဲ့ဖန်တီးသူတွေကမတူညီတဲ့နည်းလမ်းကိုရွေးချယ်လိုက်ပါ တယ်။ ။ default အနေနဲ့ Ubuntu က root account ထဲကို login ဝင်တာကိုမလုပ်ဆောင်ပေးပါဘူး။ ။ (root account ဆောက်တဲ့အခါ password ပေးမထည့်ခြင်းအားဖြင့်) ပြီးတော့ အဲဒီအစား sudo ကိုသုံးပြီးတော့ superuser privilege တွေကိုပေးအပ်ပါတယ်။ ။ အစဦးဆုံး(ဆောက်တဲ့) account ကို sudo သုံးခြင်းအားဖြင့် superuser privilege တွေကိုပေးအပ် ပြီးတော့ အဲဒါနဲ့ တူညီတဲ့ power ကိုသူ့ရဲ့ subsequent user account တွေဆီ ကိုပေးအပ်ခွင့်ရှိပါတယ်။ ။

chown—Change File Owner and Group

chown command ကို file သို့မဟုတ် directory တခုရဲ့ owner နဲ့ group owner ပြောင်းတဲ့အခါမှာ အသုံးပြုပါတယ်။ ။ ဒီ command ကိုအသုံးပြုဖို့အတွက် Superuser privilege တွေလိုအပ်ပါတယ်။ ။ chown ရဲ့ syntax ကတော့ဒီလိုပုံစံမျိုးဖြစ်ပါတယ်။ ။

chown [owner] [:[group]] file...

chown အနေနဲ့ command ရဲ့ first argument အပေါ်မူတည်ပြီးတော့ file owner နဲ့ သို့မဟုတ် file group ကိုပြောင်းပေးနိုင်ပါတယ်။ Table 9-7 မှာဥပမာအချို့ကို list လုပ်ပြထားပါတယ်။

Table 9-7: chown Argument Examples

Argument	Results
bob	file ရဲ့ပိုင်ဆိုင်မှု ownership ကို သူ့ကိုလက်ရှိသုံးနေတဲ့ current user ထံကနေ user bob ဆီကို ပြောင်းပေးလိုက်ပါတယ်။
bob:user	file ရဲ့ပိုင်ဆိုင်မှု ownership ကို သူ့ကိုလက်ရှိသုံးနေတဲ့ current user ထံကနေ user bob ဆီကို ပြောင်းပေးလိုက်ပြီးတော့ file group owner ကိုကျတော့ group users တွေထံကိုပြောင်းပေးလိုက်ပါတယ်။
admins	group owner ကို group admins တွေဆီကိုပြောင်းပေးလိုက်ပါတယ်။ file owner ကတော့မပြောင်းလဲပါဘူး။
bob:Argument	file ရဲ့ပိုင်ဆိုင်သူ owner ကို လက်ရှိ owner ထံကနေ user bob ဆီကိုပြောင်းပေးလိုက်ပြီးတော့ group owner ကိုကျတော့ user bob ရဲ့ login group ထဲကိုပြောင်းပေးလိုက်ပါတယ်။

ဆိုကြပါစို့ ကျွန်တော်တို့မှာ superuser privilege access ရှိတဲ့ Janet နဲ့ မရှိတဲ့ tony ဆိုတဲ့ user ၂ ယောက်ရှိတယ်ပေါ့။ user Janet က သူမရဲ့ home directory ထဲမှာရှိတဲ့ file တစ်ခုကို user tony ရဲ့ home directory ထဲကို copy ကူးချင်တယ်ဆိုပါတော့။ file ကို Janet က tony ကို edit လုပ်ခွင့်လည်းရရှိစေချင်တာကြောင့် Janet က copy လုပ်ထားတဲ့ file ရဲ့ ownership ကို Janet ကနေ tony ကိုပြောင်းလိုက်ပါတယ်။

```
[janet@linuxbox ~]$ sudo cp myfile.txt ~tony
Password:
[janet@linuxbox ~]$ sudo ls -l ~tony/myfile.txt
-rw-r--r-- 1 root root 8031 2012-03-20 14:30 /home/tony/myfile.txt
[janet@linuxbox ~]$ sudo chown tony: ~tony/myfile.txt
[janet@linuxbox ~]$ sudo ls -l ~tony/myfile.txt
-rw-r--r-- 1 tony tony 8031 2012-03-20 14:30 /home/tony/myfile.txt
```

ဒီမှာကျွန်တော်တို့က user Janet က file ကိုသူမရဲ့ directory ကနေ user tony ရဲ့ home directory ထဲကို copy ကူးထည့်လိုက်တာကိုတွေ့မြင်နိုင်ပါတယ်။ ပြီးတော့ Janet က file ရဲ့ ownership ကို root ကနေ (sudo သုံးလို့ဖြစ်လာတဲ့ result) tony ဆီကိုပြောင်းပေးလိုက်ပါတယ်။ trailing colon (~) ကို first argument ကိုသုံးလိုက်ခြင်းအားဖြင့် Janet က file ရဲ့ group ownership ကိုလည်းပဲ tony ရဲ့ group လည်းဖြစ်တဲ့ login group of tony ဆီကို ပြောင်းပေးလိုက်ပါတယ်။

တခုသတိထားရမှာက ပထမအကြိမ် sudo ကိုသုံးတုန်းက Janet ကိုသူမရဲ့ password ထည့်ပေးဖို့အတွက် prompt တက်တာမရှိပါဘူး။ (Ubuntu 20.04 မှာတော့ password တန်းတောင်းပါတယ်။) ဒါကဘာလို့လဲဆိုတော့ sudo ကြောင့်၊ သူ့ရဲ့ configuration တွေအများစုမှာ၊ ခင်ဗျားကို မိနစ်အနည်းငယ်လောက်အထိတော့ယုံကြည်ပေးထားပါတယ်။ (သူ့ရဲ့ timer မကုန်ခင်အထိပေါ့) (Ubuntu 20.04 မှာတော့ ခင်ဗျားကိုတစက်ကလေးမှမယုံကြည်ပေးတော့တဲ့သဘောပါပဲ :P :P)

chgrp—Change Group Ownership

Unix ရဲ့ version အဟောင်းတွေမှာ၊ chown command က file ownership တခုထဲကိုပဲပြောင်းပေးပြီးတော့ group ownership ကိုမပြောင်းပေးပါဘူး။ အဲ့ဒီအတွက်ကြောင့်မတူတဲ့နောက်ထပ် command တခု၊ chgrp ကိုသုံးပါတယ်။ သူက chown နီးပါးအလုပ်ပုံလုပ်နည်းတူပါတယ်။ သူက ကန့်သတ်ချက်တွေပိုများလာတာကလွဲရင်ပေါ့။

Exercising Your Privileges

အခုကျွန်တော်တို့က ဒီ permissions ဆိုတဲ့အရာဘယ်လိုအလုပ်လုပ်တယ်ဆိုတာကိုလေ့လာပြီးပြီဖြစ်တဲ့အတွက် အခုထုတ်ကြားဖို့အချိန်ကျပါပြီ။ ကျွန်တော်တို့ကမကြာခဏအဖြစ်များတဲ့ပြဿနာဖြစ်တဲ့ shared directory တခုကို setting လုပ်တာကို လက်တွေ့ဖြေရှင်းလုပ်ဆောင်ပြမှာဖြစ်ပါတယ်။ ကျွန်တော်တို့မှာ bill နဲ့ karen ဆိုတဲ့ user ၂ ယောက်ရှိတယ်လို့စိတ်ကူးကြည့်လိုက်ပါ။ သူတို့ ၂ ဦးစလုံးမှာ music CD collection တွေကိုယ်စီ ရှိနေကြပြီးတော့ သူတို့တဦးချင်းစီရဲ့ Ogg Vorbis သို့မဟုတ် MP3 လိုမျိုး music files တွေကိုသိမ်းထားဖို့ သူတို့က shared directory တခုလုပ်ချင်တဲ့ဆန္ဒရှိနေကြပါတယ်။ user bill မှာ sudo ကိုသုံးပြီးတော့ superuser privileges ကိုသုံးနိုင်တဲ့ access ရှိနေပါတယ်။ (user တွေဘယ်လိုဖန်တီးရသလဲဆိုတာကို ဒီမှာသွားကြည့်ပါ။

[How_to_create_a_user_on_Ubuntu_20.04](#))

ပထမဦးဆုံးလုပ်ဖို့လိုတဲ့အရာကတော့ bill ရော karen ပါ member အနေနဲ့ပါဝင်နေတဲ့ group တခုကို ဖန်တီးဖို့ပါပဲ။ GNOME's graphical user management tool ကိုအသုံးပြုပြီးတော့ bill က music လို့ခေါ်တဲ့ group တခုကိုဖန်တီးပြီးတော့ အဲ့ထဲကို user bill နဲ့ karen ကိုထည့်ပါမယ်။ (group တွေဘယ်လိုဖန်တီးရသလဲဆိုတာကို ဒီမှာသွားကြည့်ပါ။ [How to create users and group in Linux from command line](#))

***(မူရင်းစာအုပ်မှာ ubuntu version အဟောင်းပေါ်မှာ GUI နဲ့ user တွေကို group ထဲထည့်ပြထားပေမယ့် ကျွန်တော်အသုံးပြုပြီးပြမယ့် Ubuntu 20.04 မှာတော့ terminal ထဲကနေလုပ်ပြမှာဖြစ်ပါတယ်။ မရှင်းလင်းတာရှိရင်အပေါ်က link ၂ ခုမှာသေချာသွားပြန်ဖတ်ဖို့အကြံပေးလိုပါတယ်။

Terminal ထဲမှာ user အသစ်ဆောက်တာက **sudo adduser username** ဆိုတဲ့ပုံစံဖြစ်ပါတယ်။ username နေရာမှာမိမိပေးလိုတဲ့နာမည်ထည့်ပေးရမှာပါ။ အခုလေ့ကျင့်ခန်းမှာဆိုရင်တော့ bill ပေါ့။ အဲ့ဒီနေရာမှာကျွန်တော်တို့က user bill အတွက် password လည်းထည့်ချင်တဲ့အတွက်ကြောင့် adduser command ကိုသုံးတာပါ။ useradd command နဲ့လည်း user အသစ်ဆောက်လို့ရပေမယ့်သူက password မတောင်းတဲ့အတွက် GUI ကနေ setting ထဲက user မှာ user အသစ်အတွက် password ကိုသွားထည့်ပေးရင်ရပေမယ့်အလုပ်ရှုပ်ပါတယ်။

ပြီးရင် karen အတွက် account ကိုလည်းအထက်ကနည်းအတိုင်းနောက်တစ်ဆင့်ဆောက်ပေးလိုက်ပါ။ ဆောက်ထားတဲ့ user တွေကိုပြန်ဖျက်ချင်ရင်တော့ **sudo deluser - -remove-home username** ဖြစ်ပါတယ်။

```
jok3r@lucy:~$ sudo adduser bill
```

```
Adding user `bill' ...
```

```
Adding new group `bill' (1001) ...
```

```
Adding new user `bill' (1001) with group `bill' ...
```

```
The home directory `/home/bill' already exists. Not copying from `/etc/skel'.
```

```
adduser: Warning: The home directory `/home/bill' does not belong to the user you are currently creating.
```

```
New password:
```

```
Retype new password:
```

```
passwd: password updated successfully
```

```
Changing the user information for bill
```

```
Enter the new value, or press ENTER for the default
```

```
Full Name []:
```

```
Room Number []:
```

```
Work Phone []:
```

```
Home Phone []:
```

```
Other []:
```

```
Is the information correct? [Y/n] y
```

```
jok3r@lucy:~$
```

ဆောက်ပြီးတဲ့အခါသူတို့ user account တွေရှိမရှိကို စစ်ကြည့်ပါမယ်။ **sudo cat /etc/passwd** နဲ့စစ်ရပါမယ်။ user list ထွက်လာပြီးတော့ နောက်ဆုံးနားမှာအသစ်ဆောက်ထားတဲ့ bill နဲ့ karen တို့ရဲ့နာမည်တွေနဲ့သက်ဆိုင်ရာအချက်အလက်တွေကိုဖော်ပြပေးနေပါလိမ့်မယ်။ ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ user bill နဲ့ user karen တို့ရှိသွားပြီလို့ဆိုလိုပါတယ်။ GUI ကနေသွားကြည့်ချင်ရင် setting ထဲက user မှာသွားကြည့်ရင် bill နဲ့ karen တို့ရဲ့ user account ၂ ခုထပ်တိုးနေတာတွေ့ရမှာပါ။ ဒီနေရာမှာ ထွက်လာမယ့် list ကများတာကြောင့်တချို့ဟာတွေကို နဲ့ကျော်ခဲ့လိုက်ပါတယ်။

```
jok3r@lucy:~$ sudo cat /etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash
```

```
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
```

```
bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

```
....  
....  
....  
bill:x:1001:1001:,,,:/home/bill:/bin/bash  
karen:x:1002:1002:,,,:/home/karen:/bin/bash  
jok3r@lucy:~$
```

ပြီးရင်ကျွန်တော်တို့က user bill administrator rights တွေပေးပြီးတော့ karen ကိုတော့ normal user အနေနဲ့ပဲထားမှာဖြစ်ပါတယ် ။ bill ကို administrator rights တွေပေးဖို့အတွက် သူ့ကို sudo group (admin group) ထဲထည့်ရပါမယ် ။ bill ကို sudo group ထဲထည့်ဖို့အတွက် **sudo usermod -aG sudo bill** ဆိုတဲ့ command ကိုသုံးပါမယ် ။ ပြီးရင် user bill မှာ administrator rights တွေရမရကို ကြည့်ဖို့အတွက်ကျတော့ Ubuntu system မှာဆိုရင်သူက user တဦးအတွက် group တခုကို အလိုလို default တည်ဆောက်ပေးထားပါတယ် ။ ဆိုလိုတာက ခင်ဗျားအနေနဲ့ user bill ဆောက်လိုက်တာနဲ့ user bill ပါဝင်နေတဲ့ group bill ကအလိုလိုရှိပြီးသား ဖြစ်နေမှာပါ ။ ဒါ့အပြင် administrator rights တွေရဖို့ဆိုရင် user တဦးဟာ sudo group ထဲမှာပါဝင်နေဖို့လိုပါတယ် ။ ဒါကိုသဘောပေါက်ဖို့လိုပါတယ် ။ ဒါကြောင့် user bill ရဲ့ group တွေကိုကြည့်လိုက်တာနဲ့သူပါဝင်နေတဲ့ group ပေါ်မူတည်ပြီးသူမှာ administrator rights တွေရ မရဆိုတာကိုသိနိုင်ပါတယ် ။ ဒါကြောင့်ကျွန်တော်တို့က user bill ရဲ့ group တွေကို groups command နဲ့စစ်ကြည့်မှာဖြစ်ပါတယ် ။ သူပုံစံက ဒီလိုဖြစ်ပါတယ် ။

groups username

```
jok3r@lucy:~$ sudo usermod -aG sudo bill  
jok3r@lucy:~$ groups bill  
bill : bill sudo  
jok3r@lucy:~$
```

ပြီးရင်ကျွန်တော်တို့က user bill နဲ့ user karen ၂ ဦးစလုံးပါဝင်နေတဲ့ music လို့နာမည်ပေးထားတဲ့ group တခုကိုဆောက်ပါမယ် ။ ပြီးရင် user bill ကိုအရင်ဆုံးအဲ့ group ထဲထည့်ပါမယ် ။ ပြီးရင် user bill ပါဝင်နေတဲ့ group တွေကိုလည်းအထက်ကအတိုင်းပြန်စစ်ပါမယ် ။ အဲ့ဒီအခါ အောက်ကလို music ဆိုတဲ့ group တခု user bill ထဲမှာထပ်တိုးနေတာတွေ့ရပါမယ် ။ group ကိုပြန်ဖျက်ချင်ရင်တော့ **sudo delgroup groupname** ဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ sudo addgroup music
jok3r@lucy:~$ sudo usermod -aG music bill
jok3r@lucy:~$ groups bill
bill : bill sudo music
jok3r@lucy:~$
```

ပြီးရင် user karen ကိုလည်း music ဆိုတဲ့ group ထဲကိုအထက်ပါနည်းလမ်းအတိုင်းထည့်ပါမယ်။ user karen က normal user အဖြစ်ထားမှာမို့ သူ့ကို sudo group ထဲထည့်ဖို့မလိုပါဘူး။ အခုဆိုရင်ကျွန်တော်တို့မှာ administration rights တွေရထားတဲ့ user bill နဲ့ မရထားတဲ့ ရိုးရိုး normal user karen ဆိုတဲ့ user account ၂ ခု ရရှိသွားပြီဖြစ်ပါတယ်။ ပြီးရင်ကျွန်တော်တို့က user bill အနေနဲ့ login ဝင်မှာဖြစ်ပါတယ်။ အဲဒီအတွက် sudo su command ကိုအသုံးပြုပါမယ်။

```
jok3r@lucy:~$ sudo su bill
[sudo] password for jok3r:
bill@lucy:/home/jok3r$
```

အခုဆိုရင် prompt ရဲ့ username jok3r နေရာမှာ bill ဖြစ်သွားပါပြီ။ သဘောကတော့ bill အနေနဲ့ အသုံးပြုနေတဲ့သဘောပါပဲ။ bill အနေနဲ့အသုံးပြုနေရာကနေ နဂို admin ဖြစ်တဲ့ jok3r account အနေနဲ့ပြန်သုံးချင်ရင် exit လို့ရိုက်လိုက်ပါ။)***

ပြီးရင် bill က music file တွေအတွက် directory တခုဆောက်ပါမယ်။

```
bill@lucy:/home/jok3r$ sudo mkdir /usr/local/share/Music
[sudo] password for bill:
bill@lucy:/home/jok3r$
```

bill အနေနဲ့သူ့ရဲ့ home directory ပြင်ပမှာသွားပြီးတော့ file တွေကို manipulating သွားလုပ်တာဖြစ်တာကြောင့် superuser privileges တွေလိုအပ်ပါတယ်။ directory ကိုတည်ဆောက်လိုက်ပြီဆိုတာနဲ့အဲဒီ directory မှာအောက်ဖော်ပြပါ ownerships နဲ့ permissions တွေရရှိသွားပါတယ်။

```
bill@lucy:/home/jok3r$ ls -ld /usr/local/share/Music
drwxr-xr-x 2 root root 4096 ဂျူနီ  1 23:28 /usr/local/share/Music
bill@lucy:/home/jok3r$
```

ကျွန်တော်တို့မြင်နေရတဲ့အတိုင်း directory ကို root ကပိုင်ဆိုင်ထားပြီးတော့သူ့မှာ 755 permission ရှိနေပါတယ် ။ ဒီ directory ကို share လို့ရအောင်လုပ်ဖို့ဆိုရင် bill အနေနဲ့ group ownership နဲ့ group permissions ကို write လုပ်ခွင့်ရအောင် ပြောင်းပေးဖို့လိုအပ်ပါတယ် ။

```
bill@lucy:/home/jok3r$ sudo chown :music /usr/local/share/Music
bill@lucy:/home/jok3r$ sudo chmod 775 /usr/local/share/Music
bill@lucy:/home/jok3r$ ls -ld /usr/local/share/Music
drwxr-xr-x 2 root music 4096 ဂျူနီ  1 23:28 /usr/local/share/Music
bill@lucy:/home/jok3r$
```

ဒါဆို ဒါတွေအားလုံးကဘာကိုဆိုလိုတာလဲ ? သူဆိုလိုတာကအခုကျွန်တော်တို့မှာ root ကနေပိုင်ဆိုင်ထားတဲ့ /usr/local/share/Music ဆိုတဲ့ directory တခုရှိနေပြီးတော့ music ဆိုတဲ့ group ကနေ (သူ့ကို) read နဲ့ write လုပ်လို့ရတဲ့ access ရှိနေပါတယ် ။ music ဆိုတဲ့ group ထဲမှာ bill နဲ့ karen ဆိုတဲ့ member တွေရှိနေပါတယ် ။ အခြား user တွေက directory ထဲက content တွေကို list ထုတ်ကြည့်လို့ရပေမယ့် အဲဒီမှာ (directory ထဲမှာ) file တွေဆောက်ခွင့်မရှိပါဘူး ။

ဒါပေမယ့်ကျွန်တော်တို့မှာအခုအထိပြဿနာတခုရှိနေပါသေးတယ် ။ လက်ရှိ permission တွေနဲ့ဆိုရင် Music directory ထဲမှာဆောက်လိုက်တဲ့ file တွေနဲ့ directory တွေမှာ user bill နဲ့ karen တို့ရဲ့ Normal permission တွေပဲရနေပါသေးတယ် ။

```
bill@lucy:/home/jok3r$ > /usr/local/share/Music/test_file
bill@lucy:/home/jok3r$ ls -l /usr/local/share/Music/test_file
-rw-rw-r-- 1 bill bill 0 ဂျူနီ  1 23:58 /usr/local/share/Music/test_file
bill@lucy:/home/jok3r$
```

အတိအကျပြောရမယ်ဆိုရင်တော့အဲဒီမှာပြဿနာ ၂ ခု ရှိနေတာပါ ။ ပထမတခုက ဒီ system ပေါ်က umask က 0022 ဖြစ်နေတာ ၊ အဲဒါက group ထဲကအခြား member တွေပိုင်ဆိုင်တဲ့ file တွေကို group member တွေက သွားပြီး write လုပ်တာကိုပိတ်ပင်တားဆီးထားတာ ဖြစ်ပါတယ် ။ တကယ်လို့ share လုပ်ထားတဲ့ directory ထဲမှာ file တွေချည်းပဲပါရင်တော့ပြဿနာမဟုတ်ပေမယ့်လည်း directory ထဲမှာ music တွေသွားသိမ်းမှာဖြစ်ပြီးတော့ music ဆိုတာကလည်းအများအားဖြင့် artist တွေ နဲ့ album တွေအနေနဲ့ အဆင့်ဆင့်စီပြီးထားမှာဖြစ်တာ

ကြောင့် group ရဲ့ member တွေအနေနဲ့ အခြားသော member တွေတည်ဆောက်ထားတဲ့ directory တွေထဲမှာ file တွေ directory တွေတည်ဆောက်နိုင်ဖို့လိုအပ်ပါတယ် ။ ကျွန်တော်တို့အနေနဲ့ bill နဲ့ karen တို့သုံးတဲ့ umask ကို 0002 ပြောင်းပေးဖို့လိုအပ်ပါတယ် ။

ဒုတိယအချက်က member တယောက်ဆီကဖန်တီးလိုက်တဲ့ file တွေ directory တွေ တခုချင်းစီဟာ music ဆိုတဲ့ group ထက်စာရင် user ရဲ့ primary group ထဲမှာသွားထားဖို့လိုအပ်ပါတယ် ။ ဒါကို directory ပေါ်က setgid bit ကို setting ချိန်ပေးခြင်းဖြင့်ပြုပြင်လို့ရပါတယ် ။

```
bill@lucy:/home/jok3r$ sudo chmod g+s /usr/local/share/Music
[sudo] password for bill:
bill@lucy:/home/jok3r$ ls -ld /usr/local/share/Music
drwxrwsr-x 2 root music 4096 ဂျူနီ  1 23:58 /usr/local/share/Music
bill@lucy:/home/jok3r$
```

အခုဆိုရင်ကျွန်တော်တို့ကအသစ်ပေးထားတဲ့ permissions တွေကိုစမ်းသပ်ပြီးကြည့်လိုက်ပါတယ် ။ bill ကသူ့ရဲ့ umask ကို 0002 မှာထားလိုက်တယ် ၊ အရင်က test file ကိုဖျက်လိုက်ပြီးတော့ အသစ် test file နဲ့ directory ကိုတည်ဆောက်လိုက်ပါတယ် ။

```
bill@lucy:/home/jok3r$ umask 0002
bill@lucy:/home/jok3r$ rm /usr/local/share/Music/test_file
bill@lucy:/home/jok3r$ > /usr/local/share/Music/test_file
bill@lucy:/home/jok3r$ mkdir /usr/local/share/Music/test-dir
bill@lucy:/home/jok3r$ ls -l /usr/local/share/Music
total 4
drwxrwsr-x 2 bill music 4096 ဂျူနီ  2 00:22 test-dir
-rw-rw-r-- 1 bill music   0 ဂျူနီ  2 00:21 test_file
bill@lucy:/home/jok3r$ exit
jok3r@lucy:~$
```

file တွေနဲ့ directorie တွေ ၂ မျိုးစလုံးဟာ အခုဆိုရင်မှန်ကန်တဲ့ permission တွေနဲ့တည်ဆောက်လိုက်ပြီးတော့ music group ထဲက member တွေကို Music directory ထဲမှာ file တွေနဲ့ directorie တွေကို တည်ဆောက်ခွင့်ပြုလိုက်ပြီဖြစ်ပါတယ် ။

တခုထဲသောကျန်နေတဲ့ပြဿနာကတော့ umask ပဲဖြစ်ပါတယ် ။ လိုအပ်နေတဲ့ setting တွေက session မပြီးခင်အထိ (terminal မပိတ်လိုက်ခင်အထိ) ပဲခံပြီးတော့သူ့ကိုပြန်ပြန် setting ချပေးနေရပါတယ် ။ Chapter 11 ကျရင်ကျွန်တော်တို့က umask ကို permanent change လုပ်တာကို လေ့လာသွားမှာဖြစ်ပါတယ် ။

Changing Your Password

အခု Chapter မှာနောက်ဆုံးအနေနဲ့ပြောမယ့်ခေါင်းစဉ်ကတော့ မိမိကိုယ်တိုင် (နဲ့တခြား user တွေအတွက် ၊ တကယ်လို့ခင်ဗျားမှာ superuser privileges ရှိခဲ့မယ်ဆိုရင်ပေါ့) အတွက် password သတ်မှတ်ခြင်းပဲဖြစ်ပါတယ် ။ Password သတ်မှတ်တာ သို့မဟုတ် ပြောင်းဖို့ဆိုရင် passwd command ကိုအသုံးပြုပါတယ် ။ command ရဲ့ syntax ကတော့ဒီလိုပုံစံမျိုးဖြစ်ပါတယ် ။

passwd [user]

ခင်ဗျားရဲ့ password ကိုပြောင်းချင်ရင်တော့ passwd command ကိုရိုက်ရုံပါပဲ ။ (username မထည့်ရပါ) ခင်ဗျားကို password အဟောင်းနဲ့အသစ်အတွက် prompt တက်ပြီးတောင်းပါလိမ့်မယ် ။

```
jok3r@lucy:~$ passwd
Changing password for jok3r.
Current password:
New password:
```

passwd command ကနေကျွန်တော်တို့ကို strong password တွေသုံးဖို့ (ထားပေးဖို့) တွန်းအားပေးပါလိမ့်မယ် ။ ဒါက ဘာကိုဆိုလိုသလဲဆိုတော့ သူက အရမ်းတိုလွန်းတဲ့ password တွေ ၊ ပထမထားဖူးတဲ့ password တွေ နဲ့ခပ်ဆင်ဆင်တူတာတွေ ၊ dictionary ထဲကစကားလုံးတွေ သို့မဟုတ် ခန့်မှန်းရလွယ်ကူတာတွေကိုလက်ခံပေးဖို့ ငြင်းပယ်ပါလိမ့်မယ် လို့ဆိုလိုပါတယ် ။

```
jok3r@lucy:~$ passwd
Changing password for jok3r.
Current password:
New password:
Retype new password:
Password unchanged
New password:
Retype new password:
passwd: password updated successfully
jok3r@lucy:~$
```

တကယ်လို့ခင်ဗျားမှာ superuser privilege ရှိနေခဲ့မယ်ဆိုရင်တော့ ခင်ဗျားအနေနဲ့ passwd command မှာ သတ်မှတ်ထားတဲ့ username ကို argument အနေနဲ့ထားပြီးတော့အခြား user တယောက်ရဲ့ password ကို ပြောင်းပေးလို့ရပါတယ်။ superuser အနေနဲ့ account lock ချတာ၊ password သတ်တမ်းနဲ့ပက်သက်ပြီးသတ်မှတ် တာ စသည်တွေအတွက် အခြားသော option တွေ ရှိပါသေးတယ်။ passwd man page မှာအသေးစိတ်သွားဖတ် ကြည့်လိုက်ပါ။

10 PROCESSES

ခေတ်သစ် operating system တွေဟာထုံးစံအတိုင်း multitasking တွေဖြစ်ပါတယ်။ ဆိုလိုတာက သူတို့ ဟာ တကြိမ်ထဲမှာတစ်ခုထက်ပိုတဲ့အလုပ်တွေလုပ်နိုင်တယ်လို့ထင်မှတ်မှားလာအောင် executing program တခုက နေ နောက်တခုဆီကိုဆက်တိုက်လျင်မြန်စွာပြောင်းရွှေ့လုပ်ကိုင်တာမျိုးကိုဖန်တီးနိုင်တာကိုဆိုလိုပါတယ်။ ဒါကို Linux kernel က process တွေကိုအသုံးပြုပြီးတော့ကိုင်တွယ်ပါတယ်။ process တွေဆိုတာကတော့မတူညီတဲ့ program တွေက CPU ထံမှာသူတို့အလှည့်ကျအောင်စောင့်ဆိုင်းတာကို Linux ကဘယ်လို စည်းစနစ်နဲ့လုပ်ထား တယ်ဆိုတာပါပဲ။

တခါတရံမှာ computer ကနွေးကွေးလေးလံလာတာ သို့မဟုတ် application တခုကတုန့်ပြန်မှုမလုပ်ပေး တော့တာ တွေဖြစ်တတ်ပါတယ်။ အခု Chapter မှာ ကျွန်တော်တို့ကို program တွေဘာလုပ်နေသလဲဆိုတာကို စစ်ဆေးခွင့်ပေးပြီးတော့ မူမှန်တဲ့ process တွေကိုပိတ်ချခွင့်ပေးတဲ့ ကျွန်တော်တို့အနေနဲ့ command line မှာရနိုင် တဲ့ tool အချို့ကို လေ့လာသွားမှာဖြစ်ပါတယ်။

အခု Chapter မှာအောက်ပါ command တွေကိုမိတ်ဆက်ပေးသွားမှာဖြစ်ပါတယ်။

- **ps** — လက်ရှိအလုပ်လုပ်နေတဲ့ process ရဲ့ snapshot တခုကို report ထုတ်ပေးပါတယ်။
- **top** — task တွေကိုဖော်ပြပေးပါတယ်။
- **jobs** — active ဖြစ်နေတဲ့ job တွေကိုဖော်ပြပေးပါတယ်။
- **bg** — job တခုကို background ကိုပို့ပေးပါတယ်။

- **fg** — job တခုကို foreground ကိုပို့ပေးပါတယ် ။
- **kill** — process တခုဆီကို signal တခုပို့ပေးပါတယ် ။
- **killall** — process တွေကိုနာမည်နဲ့တကွ kill လုပ်ပေးပါတယ် ။
- **shutdown** — system ကြီးတခုလုံးကို shutdown သို့မဟုတ် reboot လုပ်ပေးပါတယ် ။

How a Process Works

system တခုစတင်တဲ့အခါမှာ kernel ကသူ့ကိုယ်ပိုင်လုပ်ဆောင်ချက်အချို့ကို process အနေနဲ့စတင်ပေးပြီးတော့ init လို့ခေါ်တဲ့ program တခုကိုဖွင့်လိုက်ပါတယ် ။ init ရဲ့အလှည့်မှာ init scripts လို့ခေါ်တဲ့ shell script တွေ (/etc ထဲမှာရှိတယ်) ကိုအစဉ်လိုက် run ပေးပြီးတော့အဲ့ဒါက system services တွေအကုန်လုံးကိုစတင်ပေးလိုက်ပါတယ် ။ ဒီ service တွေအများစုဟာ daemon programs တွေအနေနဲ့ implement လုပ်ပေးပါတယ် ။ (daemon programs တွေဆိုတာ) background မှာထိုင်နေတဲ့ program တွေဖြစ်ပြီး user interface မရှိပဲသူတို့အလုပ်သူတို့လုပ်နေကြတာဖြစ်ပါတယ် ။ ဒါကြောင့်ကျွန်တော်တို့အနေနဲ့ login မဝင်ထားခဲ့ရင်တောင်မှ system ဟာ သူ့ပုံမှန်လမ်းကြောင်းအတိုင်းလုပ်စရာရှိတာတွေလုပ်ရင်းအနည်းငယ်အလုပ်ရှုပ်နေတတ်ပါတယ် ။

တကယ်တမ်းက program တခုကအခြား program တခုကိုဖွင့်ပေးနိုင်တယ်ဆိုတာကို process scheme မှာကျတော့ parent process တခုက child process တခုကိုထုတ်လုပ်ပေးတယ်လို့ဖော်ပြပါတယ် ။

kernel က process တခုချင်းစီရဲ့သတင်းအချက်အလက်တွေကိုထိန်းသိမ်းပေးထားခြင်းအားဖြင့်အရာရာကိုစုစည်းစည်းရှိစေဖို့အကူအညီပေးပါတယ် ။ ဥပမာ - process တခုချင်းစီကို process ID (PID) လို့ခေါ်တဲ့နံပါတ်တခုစီ assign ထည့်ပေးထားပါတယ် ။ PID တွေဟာ ငယ်စဉ်ကြီးလိုက်ပုံစံနဲ့စီထားပြီးတော့ init ကိုအမြဲတမ်း PID 1 အဖြစ်ထားပါတယ် ။ kernel က process တခုချင်းစီကို memory assign လုပ်တာတွေနဲ့အတူ process တွေ resume execution လုပ်ဖို့အတွက်အဆင်သင့်ဖြစ်မဖြစ်တွေကိုပါ ခြေရာခံပေးပါတယ် ။ file တွေလိုပဲ process တွေမှာလည်း effective user ID တွေအစရှိသည်ဖြင့် owner ပိုင်ရှင်နဲ့ user ID တွေရှိပါတယ် ။

Viewing Processes with ps

process တွေကိုကြည့်ဖို့အတွက်အသုံးအများဆုံး command (အခြားအများကြီးရှိပါသေးတယ်) ကတော့ ps ဖြစ်ပါတယ် ။ ps program မှာ option တွေအများကြီးရှိပါတယ် ။ ဒါပေမယ့်သူ့ရဲ့အရိုးရှင်းဆုံးအသုံးပြုတဲ့ပုံစံကတော့ဒီလိုဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ ps
  PID TTY          TIME CMD
 40380 pts/0    00:00:00 bash
 40407 pts/0    00:00:00 ps
jok3r@lucy:~$
```

အခုပမာထဲက result ထဲမှာ process ၂ ခုကို list ထုတ်ပြပေးပါတယ်။ process 40380 နဲ့ process 40407 တို့ဖြစ်ကြပြီးတော့သူတို့က bash နဲ့ ps တို့အစဉ်အတိုင်းဖြစ်ပါတယ်။ ကျွန်တော်တို့အခုမြင်နေရတဲ့အတိုင်း ps က default အနေနဲ့ကျွန်တော်တို့ကိုသိပ်မပြထားပါဘူး။ လက်ရှိ terminal session နဲ့ပက်သက်တဲ့ process လောက်ပဲပြထားပေးပါတယ်။ (အဲ့ဒါထက်) ပိုကြည့်ဖို့အတွက်ကျွန်တော်တို့အနေနဲ့ option အချို့ထပ်ထည့်ပေးဖို့လိုပါမယ်။ ဒါပေမယ့်အဲ့ဒါကိုကျွန်တော်တို့မလုပ်ခင်မှာ ps က ထုတ်ပေးတဲ့အခြား field တွေကိုတချက်အရင်ကြည့်ကြစို့။ TTY ဆိုတာက teletype ရဲ့အတိုကောက်ဖြစ်ပြီးတော့ process အတွက် terminal ကိုထိန်းချုပ်ခြင်းလို့ရည်ညွှန်းပါတယ်။ Unix ကဒီနေရာမှာ သူရဲ့သက်တမ်းကိုပြနေတာပါပဲ။ TIME field ကတော့ process ကသုံးထားတဲ့ CPU time ပမာဏ ကိုပြနေတာပါပဲ။ ကျွန်တော်တို့အခုမြင်နေရတဲ့အတိုင်း process ၂ခုစလုံးက computer ကိုအလုပ်တွေအရမ်းကြိုးစားခိုင်းထားပါတယ်။ တကယ်လို့ကျွန်တော်တို့အနေနဲ့ option ထည့်ခဲ့မယ်ဆိုရင်ကျွန်တော်တို့အနေနဲ့ system ကဘာတွေလုပ်နေသလဲဆိုတာကိုပိုကြီးမားတဲ့မြင်ကွင်းတခုအဖြစ်မြင်ရမှာဖြစ်ပါတယ်။

jok3r@lucy:~\$ ps x

PID	TTY	STAT	TIME	COMMAND
1318	?	Ss	0:03	/lib/systemd/systemd --user
1320	?	S	0:00	(sd-pam)
1330	?	Ssl	0:00	/usr/bin/pulseaudio --daemonize=no --log-target=jou
1332	?	SNsl	0:02	/usr/libexec/tracker-miner-fs
1335	?	SLl	0:00	/usr/bin/gnome-keyring-daemon --daemonize --login
1339	?	Ss	0:03	/usr/bin/dbus-daemon --session --address=systemd: -
1354	tty2	Ssl+	0:00	/usr/lib/gdm3/gdm-wayland-session env GNOME_SHELL_S
1365	tty2	Sl+	0:00	/usr/libexec/gnome-session-binary --systemd --syste
1375	?	Ssl	0:00	/usr/libexec/gvfsd
...				
...				

စသည်ဖြင့်နောက်ထပ်များစွာဆက်ရှိနေပါသေးတယ်

x option ကိုထည့်လိုက်ခြင်းအားဖြင့် (x ရဲ့အရှေ့မှာ - မပါတာသတိပြုပါ) ps ကို terminal ကထိန်းချုပ်ထားတာ (တကယ်လို့ရှိခဲ့ရင်) ကလွဲရင်ကျန်တဲ့ကျွန်တော်တို့ရဲ့ process အကုန်လုံးကိုပြပေးဖို့ပြောပေးပါတယ်။ TTY column ထဲက ? အမှတ်အသားက ထိန်းချုပ်ထားတဲ့ terminal မရှိကြောင်းပြတာဖြစ်ပါတယ်။ ဒီ option ကိုအသုံးပြုခြင်းအားဖြင့် ကျွန်တော်တို့အနေနဲ့ ကျွန်တော်တို့ပိုင်ဆိုင်တဲ့ process တိုင်းရဲ့ list ကို မြင်ရမှာဖြစ်ပါတယ်။

system မှာ process တွေအများကြီး run နေတာကြောင့် ps က list အရှည်ကြီးကိုထုတ်ပေးခြင်းဖြစ်ပါတယ်။ ps ရဲ့ output တွေကို pipe နဲ့ less ထဲကို ပို့ပြီးပိုပြီးကြည့်ရလွယ်အောင်မကြာခဏလုပ်ခြင်းကအကူအညီဖြစ်စေပါတယ်။ တချို့ option တွေကိုပေါင်းစပ်ခြင်းကလည်း output တွေရဲ့ list ကိုရှည်လျားစေတာကြောင့် terminal emulator window ကိုအကျယ်ချဲ့ထားခြင်းကလည်းကောင်းတဲ့အကြံတခုဖြစ်ပါတယ်။

STAT လို့ခေါင်းစဉ်တပ်ထားတဲ့ column အသစ်တခုကို output မှာထပ်ပေါင်းထည့်ထားပါတယ်။ STAT ဆိုတာ state ရဲ့အတိုကောက်ဖြစ်ပြီးတော့process ရဲ့ လက်ရှိ status ကို Table 10-1 မှာပြထားသလိုဖော်ပြပေးပါတယ်။

Table 10-1: Process States

State	Meaning
R	Running ဖြစ်နေတာပါ။ Process က running ဖြစ်နေတာ သို့မဟုတ် run ဖို့အဆင်သင့်ဖြစ်နေတာပါ။
S	Sleeping ဖြစ်နေတာပါ။ Process က running ဖြစ်နေပေမယ့် keystroke တချက် သို့မဟုတ် network packet လိုဟာမျိုး event တခုကိုစောင့်ဆိုင်းနေတာပါ။
D	(သူ့ကို)အနှောက်အယှက်ပေးလို့မရပဲ sleep နေတာပါ။ Process က hard disk လိုမျိုး I/O တခုကိုစောင့်ဆိုင်းနေတာပါ။
T	Stop ဖြစ်နေတာပါ။ Process က ရပ်ဖို့အတွက်ညွှန်ကြားခြင်းခံထားရပါတယ်။ (ဒါကိုနောက်မှာထပ်ပြီးပြောပါမယ်)
Z	defunct တခု သို့မဟုတ် “zombie” process ဖြစ်နေတာပါ။ ဒါက terminate လုပ်ခံထားရတဲ့ child process တခုပါ ဒါပေမယ့်သူ့ရဲ့ parent process က clean up လုပ်မပေးသေးတာပါ။
<	high-priority process တခုဖြစ်ပါတယ်။ ဒါက process တခုကိုအလေးပေးမှုတခုကိုပေးအပ်ခြင်းခံထားရတာလည်းဖြစ်နိုင်ပါတယ်။ CPU ပေါ်မှာအချိန်ပိုပေးထားတာမျိုးပါ။ process တခုရဲ့ဒီလိုပိုင်ဆိုင်ထားမှုကို niceness လို့ခေါ်ပါတယ်။ high priority နဲ့ process တွေက niceness နည်းတယ်လို့ပြောလေ့ရှိတာက သူက CPU ရဲ့အချိန်ကိုပိုယူထားတာကြောင့်အခြားလူတွေအကုန်လုံးအတွက် (CPU ပေါ်ကအချိန်ရရှိမှု) အချိန်နည်းသွားစေတာကြောင့်ဖြစ်ပါတယ်။
N	low-priority process တခုဖြစ်ပါတယ်။ low priority (nice process တခု) နဲ့ process တခုဟာ higher priority နဲ့ process တွေကို (CPU က) service ပေးလို့ပြီးတဲ့အချိန်ကျမှသာ processor ရဲ့ အချိန်ကိုရရှိကြပါတယ်။

အဲ့ဒီ process တွေရဲ့ state တွေနောက်မှာအခြား character တွေပါလာကောင်းပါလားနိုင်ပါတယ်။ ဒါတွေကအမျိုးအစားများပြားလှတဲ့ exotic process characteristic တွေကိုညွှန်ပြနေပါတယ်။ အသေးစိတ်အချက်အလက်တွေကို ps man page မှာသွားဖတ်လိုက်ပါ။

နောက်ထပ်ကျော်ကြားလှတဲ့ option set တခုကတော့ aux (သူ့အရှေ့မှာ - မပါ) ဖြစ်ပါတယ်။ ဒါကကျွန်တော်တို့ကို အချက်အလက်တွေပိုပြီးတောင်ပေးပါသေးတယ်။

```
jok3r@lucy:~$ ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.3	0.1	168344	12224	?	Ss	19:02	0:06	/sbin/init sp
root	2	0.0	0.0	0	0	?	S	19:02	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	19:02	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	19:02	0:00	[rcu_par_gp]

```

root      6   0.0   0.0       0   0 ?      I<  19:02   0:00 [kworker/0:0H
root      9   0.0   0.0       0   0 ?      I<  19:02   0:00 [mm_percpu_wq
root     10   0.0   0.0       0   0 ?      S   19:02   0:00 [rcu_tasks_ru
root     11   0.0   0.0       0   0 ?      S   19:02   0:00 [rcu_tasks_tr
...

```

စသည်ဖြင့်နောက်ထပ်များစွာဆက်ရှိနေပါသေးတယ်

ဒီ option set တွေက user တိုင်းနဲ့သက်ဆိုင်တဲ့ process တွေကိုဖော်ပြပေးနေပါတယ် ။ option တွေရဲ့အရှေ့မှာ - မ ပါပဲ အသုံးပြုခြင်းဖြင့် command ကို BSD-style အပြုအမူမျိုး invoke ပြုလုပ်ပေးပါတယ် ။ Linux version က ps ဟာ Unix implementation တွေအများအပြားမှာတွေ့နိုင်တဲ့ ps program ကိုအတုယူပြီးသူနဲ့အလားတူလုပ်ဆောင်ပေးနိုင်ပါတယ် ။ ဒီ option တွေရဲ့ဆိုရင်ကျွန်တော်တို့မှာနောက်တိုး column တွေ Table 10-2 မှာပြထားသလိုရရှိလာမှာဖြစ်ပါတယ် ။

Table 10-2: BSD-Style ps Column Headers

Header	Meaning
USER	User ID ဖြစ်ပါတယ် ။ ဒါက process ရဲ့ owner လည်းဖြစ်ပါတယ် ။
%CPU	CPU အသုံးပြုမှုနှုန်းကို ရာခိုင်နှုန်းအနေနဲ့ပြပေးပါတယ် ။
%MEM	Memory အသုံးပြုမှုနှုန်းကို ရာခိုင်နှုန်းအနေနဲ့ပြပေးပါတယ် ။
VSZ	Virtual memory size ဖြစ်ပါတယ် ။
RSS	Resident Set Size ဖြစ်ပါတယ် ။ process ကအသုံးပြုနေတဲ့ physical memory (RAM) အရေအတွက်ကို kilobyte နဲ့ပြပေးပါတယ် ။
START	process စတဲ့အချိန်ကနေစမှတ်ပေးတာဖြစ်ပါတယ် ။ ၂၄ နာရီကျော်သွားတဲ့တန်ဖိုးတွေအတွက် တရက်အဖြစ်သတ်မှတ်အသုံးပြုပါတယ် ။

Viewing Processes Dynamically with top

ps command အနေနဲ့စက်ကဘာတွေလုပ်နေလဲဆိုတာကိုအများကြီးဖော်ထုတ်ပေးနေစဉ်မှာသူက ps command ကို execute လုပ်နေတဲ့အချိန် စက်ရဲ့ state ကို snapshot တခုထဲအနေနဲ့ ပြုလုပ်ပေးနိုင်ပါတယ် ။ စက်ရဲ့ activity တွေကို dynamic view အနေနဲ့ပိုမိုကြည့်ရှုနိုင်ဖို့ရာကျွန်တော်တို့က top command ကိုအသုံးပြုပါတယ် ။

top program ဟာ ဆက်တိုက်ဆိုသလို updating လုပ်ပြီး (default အရဆိုရင် ၃ စက္ကန့်တိုင်းမှာပေါ့)ဖော်ပြပေးနေပါတယ် ။ process activity အပေါ်မူတည်ပြီးစီထားတဲ့ system processe တွေကိုဖော်ပြပေးနေတာဖြစ်ပါတယ် ။ သို့နာမည်က system ပေါ်က top process တွေကိုကြည့်ဖို့အတွက် top program ကိုသုံးတယ်ဆိုတဲ့ဖြစ်ရပ်မှန်ကနေယူထားတာဖြစ်ပါတယ် ။ top ကနေဖော်ပြပေးတာမှာအပိုင်း ၂ ပိုင်းပါဝင်နေပါတယ် ။ ထိပ်ဆုံးမှာဖော်ပြမှုမှာ system summary တခုဖော်ပြထားပေးပြီး တော့ CPU activity အပေါ်မူတည်ပြီးတော့စီထားတဲ့ process table တခုသို့နောက်မှာတွဲလျက်ကပ်ပါပါတယ် ။

jok3r@lucy:~\$ top

```
jok3r@lucy:~$ top
top - 16:42:10 up 1:48, 1 user, load average: 0.37, 0.41, 0.27
Tasks: 267 total, 1 running, 266 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.0 us, 0.9 sy, 0.0 ni, 95.9 id, 1.2 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7839.3 total, 4877.2 free, 1141.6 used, 1820.5 buff/cache
MiB Swap: 2048.0 total, 2048.0 free, 0.0 used, 6276.0 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 1662 jok3r    20   0 5294628 324520 123472 S   6.3   4.0   2:08.31 gnome-s+
 2281 jok3r    20   0 311260    8284   6760 S   1.3   0.1   0:11.78 ibus-da
 5101 jok3r    20   0 1167228 335628 141880 S   1.3   4.2   1:45.78 soffice+
    1 root       20   0 168044    11684   7992 S   0.7   0.1   0:13.54 systemd
 2287 jok3r    20   0 345324    26576   17380 S   0.7   0.3   0:02.78 ibus-ex+
 20036 jok3r    20   0 12288    3892   3128 R   0.7   0.0   0:00.09 top
    13 root       20   0      0      0      0 I   0.3   0.0   0:02.39 rcu_sch+
 1096 systemd+  20   0 23868    12008   8084 S   0.3   0.1   0:04.93 systemd+
 1191 message+  20   0 9804     6176   3836 S   0.3   0.1   0:10.67 dbus-da
 2340 jok3r    20   0 163128    7008   6376 S   0.3   0.1   0:03.64 ibus-en+
 17530 root       20   0      0      0      0 I   0.3   0.0   0:00.35 kworker+
 20003 jok3r    20   0 401460   49548   38624 S   0.3   0.6   0:00.54 gnome-t+
    2 root       20   0      0      0      0 S   0.0   0.0   0:00.01 kthreadd
    3 root       0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_gp
    4 root       0 -20      0      0      0 I   0.0   0.0   0:00.00 rcu_par+
    6 root       0 -20      0      0      0 I   0.0   0.0   0:00.00 kworker+
    9 root       0 -20      0      0      0 I   0.0   0.0   0:00.00 mm_perc+

jok3r@lucy:~$ ^C
jok3r@lucy:~$
```

system summary ထဲမှာအသုံးဝင်တဲ့အရာတွေအများကြီးပါဝင်ပါတယ် ။ အတွက် Table 10-3 ကိုကြည့်လိုက်ပါ ။

Table 10-3: top Information Fields

Row	Field	Meaning
1	top	program ရဲ့နာမည်ဖြစ်ပါတယ် ။
	16:42:10	လက်ရှိရက်(ယနေ့)ရဲ့အချိန် ဖြစ်ပါတယ် ။
	up 1:48	ဒါကို uptime လို့ခေါ်ပါတယ် ။ စက်ကနောက်ဆုံးအကြိမ် boot လုပ်ပြီးထဲကအခုအထိ ရှိနေတဲ့

		အချိန်အတိုင်းအတာဖြစ်ပါတယ်။ ။ အခုဥပမာမှာဆိုရင် system စတင်ခဲ့တာ ၁ နာရီနဲ့ ၄၈ မိနစ်ရှိနေပြီဖြစ်ပါတယ်။ ။
	1 user	user ၁ ယောက်ပဲ log in ဝင်ထားပါတယ်။ ။
	load average:	Load average ဆိုတာ run ဖို့စောင့်နေတဲ့ process အရေအတွက်ကိုရည်ညွှန်းပါတယ်။ ။ ဒါပါပဲ သူတို့ဟာ run လို့ရတဲ့ state တခုထဲမှာရောက်နေပြီးတာ့ CPU ကိုမျှဝေသုံးစွဲနေကြတဲ့ program အရေအတွက်ကိုပြောတာပါပဲ။ ။ တန်ဖိုး ၃ ခုကိုပြထားပြီးတော့တခုစီကမတူညီတဲ့ အချိန်ကာလတခုစီကိုပြနေပါတယ်။ ။ ပထမတခုကလွန်ခဲ့တဲ့စက္ကန့် ၆၀ အတွက် average ဖြစ်ပါတယ်။ ။ သူ့နောက်ကဟာက လွန်ခဲ့တဲ့ ၅ မိနစ် နဲ့နောက်ဆုံးတခုက လွန်ခဲ့တဲ့ ၁၅ မိနစ်အတွက် ဖြစ်ပါတယ်။ ။ 1.0 အောက်တန်ဖိုးတွေကတော့စက်ကအလုပ်များမနေဘူးလို့ပြနေတာဖြစ်ပါတယ်။ ။
2	Tasks:	ဒါက processes အရေအတွက်နဲ့သူတို့ရဲ့ process state မျိုးစုံကိုစုပေါင်းပြသပေးပါတယ်။ ။
	2.0 us	user processe တွေအတွက် CPU ရဲ့ 2.0% ကိုအသုံးပြုနေကြောင်းပြနေပါတယ်။ ။ ဒါက kernel ရဲ့အပြင်ဖက်က process တွေကိုဆိုလိုပါတယ်။ ။
	0.9 sy	system (kernel) process တွေအတွက် CPU ရဲ့ 0.9% ကိုအသုံးပြုနေကြောင်းပြနေပါတယ်။ ။
	0.0 ni	nice (low-priority) process တွေအတွက် CPU ရဲ့ 0.9% ကိုအသုံးပြုနေကြောင်းပြနေပါတယ်။ ။
	95.9 id	CPU ရဲ့ 95.9% ဟာအားလပ်နေပါတယ်။ ။
	1.2 wa	CPU ရဲ့ 1.2% ဟာ I/O ကိုစောင့်ဆိုင်းနေပါတယ်။ ။
	0.0 hi	hardware irq သို့မဟုတ် hardware interrupt ဖြစ်နေစဉ်မှာ servicing/handling လုပ်ဖို့အတွက် CPU ကသုံးပေးတဲ့အချိန် %
	0.0 si	software irq သို့မဟုတ် software interrupt ဖြစ်နေစဉ်မှာ servicing/handling လုပ်ဖို့အတွက် CPU ကသုံးပေးတဲ့အချိန် %
	0.0 st	steal time --- hypervisor ကနောက်ထပ် process တခုကို servicing လုပ်နေစဉ် (သို့မဟုတ်) virtual machine ထဲကနေ CPU ရဲ့အချိန် % ခိုးယူခံရနေစဉ်မှာ virtual CPU က စောင့်ဆိုင်းနေတဲ့ involuntary ထဲက CPU ရဲ့အချိန် %
	Mem:	physical RAM ဘယ်လောက်သုံးထားသလဲဆိုတာကိုပြနေတာပါ။ ။
	Swap	swap space (virtual memory) ကိုဘယ်လိုသုံးနေသလဲဆိုတာကိုပြပေးပါတယ်။ ။

top program က keyboard command တွေကိုအရေအတွက်တခုအထိလက်ခံပေးပါတယ်။ ။ စိတ်ဝင်စားစရာအကောင်းဆုံး ၂ ခုကတော့ program ရဲ့ help screen ကိုဖော်ပြပေးတဲ့ h နဲ့ top ထဲကနေထွက်ပေးတဲ့ q တို့ပဲ ဖြစ်ပါတယ်။ ။

major desktop environment ၂ ခုလုံးမှာ top လိုမျိုး (windows ထဲမှာ task manager အလုပ်လုပ်ပေးတဲ့ပုံစံလိုမျိုးပဲ) information တွေဖော်ပြပေးတဲ့ graphical application တွေထည့်သွင်းပေးထားပေမယ့် ကျွန်တော့်

အနေနဲ့ကတော့ graphical version တွေထက်စာရင် top ကိုပိုကြိုက်ပါတယ် ။ ဘာလို့လဲဆိုတော့သူက ပိုမြန်ပြီး တော့ system resource စားတာသက်သာလို့ပါပဲ ။ တကယ်တော့ ကျွန်တော်တို့ရဲ့ system monitor program ဟာ ကျွန်တော်တို့ခြေရာခံဖို့လုပ်နေတဲ့ system slowdown (system ကိုနှေးကွေးစေတဲ့) အထဲမှာပါဝင်နေသင့်ပါဘူး ။

Controlling Processes

အခုကျွန်တော်တို့က process တွေကိုတွေ့မြင်ပြီး monitor (စောင့်ကြည့်တာ) လည်းလုပ်ပြီးပြီဆိုတော့ သူတို့အပေါ်မှာထိန်းချုပ်မှု control အနည်းငယ်ရအောင်လုပ်ကြည့်ကြစို့ ။ ကျွန်တော်တို့ရဲ့ စမ်းသပ်မှုအတွက် ကျွန်တော်တို့က xlogo လို့ခေါ်တဲ့ program အသေးလေးတခုကိုအစမ်းသပ်ခံလေးအဖြစ်အသုံးပြုမှာဖြစ်ပါတယ် ။ xlogo program ဆိုတာ X Window System (ကျွန်တော်တို့ရဲ့စက်ပေါ်က display တွေမှာ graphic တွေမြင်ရအောင်နောက်ကွယ်ကနေအလုပ်လုပ်ပေးနေတဲ့ engine) ထဲမှာတပါထဲထည့်ပေးထားတဲ့ sample program လေး ဖြစ်ပြီးတော့သူ့မှာ x logo လေးပါတဲ့ ချုံ့လို့ချဲ့လို့ရတဲ့ windows လေးပါဝင်ပါတယ် ။ ပထမဆုံးကျွန်တော်တို့က ကျွန်တော်တို့စမ်းသပ်မယ့်အစမ်းသပ်ခံနဲ့ရင်းနှီးအောင်လုပ်ပါမယ် ။

jok3r@lucy:~\$ xlogo

ကျွန်တော်တို့က command ကိုရိုက်ထည့်လိုက်တာနဲ့ logo ပါတဲ့ windows အသေးလေးတခုက screen ရဲ့တနေရာရာမှာပေါ်လာမှာဖြစ်ပါတယ် ။ တချို့ system တွေပေါ်မှာဆိုရင်တော့ xlogo ကို သတိပေး message အနေနဲ့ပြနေပေမယ့်ခင်ဗျားအနေနဲ့လုံခြုံစိတ်ချစွာ(အဲ့ဒီ message ကို) ဂရုမစိုက်ပဲပိတ်လိုက်လို့ရပါတယ် ။

Note: တကယ်လို့ခင်ဗျားရဲ့ system ပေါ်မှာ xlogo program မပါလာခဲ့ရင် သူ့အစား gedit သို့မဟုတ် kwrite နဲ့ စမ်းသပ်လို့လည်းရပါတယ် ။

ကျွန်တော်တို့အနေနဲ့ xlogo run နေကြောင်းကိုသူ့ရဲ့ windows size ကိုချုံ့ချဲ့ပြီးပြောင်းကြည့်ခြင်းဖြင့်သိနိုင်ပါတယ် ။ တကယ်လို့ logo ကခင်ဗျားအသစ်ပြောင်းလိုက်တဲ့ windows size အတိုင်းလိုက်ပြောင်းပေးတယ်ဆိုရင် program က run နေတယ်လို့ယူဆလို့ရပါတယ် ။

ကျွန်တော်တို့ရဲ့ shell prompt ဘယ်လိုဖြစ်လို့ပြန်မလာတာလဲဆိုတာကိုသတိထားမိလား ? ဒါကဘာလို့လဲဆိုတော့ shell prompt က ကျွန်တော်တို့အခုအထိအသုံးပြုခဲ့ဖူးပြီးဖြစ်တဲ့တခြား program တွေလိုပဲ program ပြီးတဲ့အထိစောင့်နေလို့ဖြစ်ပါတယ် ။ တကယ်လို့ကျွန်တော်တို့က xlogo window ကိုပိတ်လိုက်မယ်ဆိုရင်တော့ prompt (terminal) ဆီပြန်ရောက်သွားမှာဖြစ်ပါတယ် ။

Interrupting a Process

ကျွန်တော်တို့ xlogo ကို run လိုက်တဲ့အခါမှာဘာတွေဖြစ်သွားသလဲဆိုတာကိုသေချာစေ့စပ်စွာကြည့်ကြပါစို့ ။ ပထမဆုံးအနေနဲ့ xlogo command ကိုရိုက်ထည့်လိုက်ပြီးတော့ program က running ဖြစ်မဖြစ်ကြည့်လိုက်ပါ ။ ပြီးရင် (သူ့ကိုမပိတ်ပဲ) terminal window ကိုပြန်သွားပြီးတော့ Ctrl + C ကိုနှိပ်လိုက်ပါ ။

```
jok3r@lucy:~$ xlogo
jok3r@lucy:~$
```

terminal တခုအတွင်းမှာ Ctrl + C နှိပ်လိုက်ခြင်းဟာ program တခုကို interrupt ကြားဖြတ်ရပ်တန့်စေပါတယ် ။ ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ကျွန်တော်တို့က program ကိုပိတ်ပစ်ဖို့ယဉ်ကျေးစွာတောင်းဆိုလိုက်တာ ဖြစ်ပါတယ် ။ ကျွန်တော်တို့က Ctrl + C နှိပ်လိုက်ပြီးတဲ့နောက်မှာတော့ xlogo window ပိတ်သွားပြီးတော့ shell prompt ပြန်ရောက်လာပါတယ် ။

များစွာသော (အကုန်လုံးတော့မဟုတ်ပါ) command-line program တွေကို ဒီနည်းလမ်းကိုအသုံးပြုပြီး တော့interrupt ကြားဖြတ်ရပ်တန့်စေနိုင်ပါတယ် ။

Putting a Process in the Background

ကျွန်တော်တို့အနေနဲ့ xlogo program ကိုပိတ်မပြစ်ပဲ shell prompt ပြန်ရောက်လာစေချင်တယ်ဆိုပါစို့ ။ ဒါကိုကျွန်တော်တို့က program ကို background ကိုပို့လိုက်ခြင်းအားဖြင့်လုပ်နိုင်ပါတယ် ။ terminal မှာ foreground (shell prompt လိုမျိုး မျက်နှာပြင်ပေါ်မှာမြင်တွေ့လို့ရတဲ့အရာတွေနဲ့ဟာမျိုး) တခုနဲ့ background (မျက်နှာပြင်ရဲ့အောက်မှာဖွက်ထားတဲ့အရာတွေနဲ့ဟာမျိုး) တခုရှိတယ်လို့တွေးကြည့်လိုက်ပါ ။ program တခုကိုဖွင့် လိုက်ပြီးတော့သူ့ကိုချက်ချင်းပဲ background ကိုရောက်သွားစေချင်တယ်ဆိုရင် ကျွန်တော်တို့က command ရဲ့ အနောက်မှာ (&) character တခုထည့်ပေးရပါမယ် ။

```
jok3r@lucy:~$ xlogo &
[1] 18047
jok3r@lucy:~$
```

command ကိုရိုက်ထည့်လိုက်ပြီးတဲ့အခါမှာ xlogo window ပေါ်လာပြီးတော့ shell prompt ဆီကိုပြန် ရောက်သွားပြီး (Ubuntu 20.04 မှာတော့shell prompt ဆီကိုပြန်မသွားပါဘူး) သူ့မှာဘာမှန်းမသိတဲ့ နံပါတ်တွေပါ ပေါ်နေပါလိမ့်မယ် ။ ဒီ message ဟာ shell feature ရဲ့အစိတ်အပိုင်းတခုဖြစ်ပြီးတော့သူ့ကို job control လို့ခေါ်ပါ တယ် ။ အဲ့ဒီ message နဲ့ shell ကကျွန်တော်တို့ဟာ job number 1 ([1]) နဲ့ PID 18047 ကိုစတင်ထားတယ်လို့ ကျွန်တော်တို့ကိုပြောနေတာဖြစ်ပါတယ် ။ ကျွန်တော်တို့ ps ကို run လိုက်မယ်ဆိုရင် ကျွန်တော်တို့ရဲ့ process တွေကိုမြင်နေရ မှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ ps
  PID  TTY      TIME  CMD
 18596 pts/0    00:00:00 bash
 18618 pts/0    00:00:00 xlogo
```

jok3r@lucy:~\$

shell ရဲ့ job control facility ကလည်းပဲ ကျွန်တော်တို့ရဲ့ terminal ကနေဖွင့်ထားတဲ့ job တွေကို list လုပ်ကြည့်ဖို့ အတွက်နည်းလမ်းတခုကျွန်တော်တို့ကိုပေးပါတယ် ။ jobs command ကိုအသုံးပြုခြင်းဖြင့်ကျွန်တော်တို့အနေနဲ့အောက်ပါ list ကိုမြင်တွေ့နိုင်မှာဖြစ်ပါတယ် ။

jok3r@lucy:~\$ jobs

[1]+ Running xlogo &

jok3r@lucy:~\$

result တွေမှာ ကျွန်တော်တို့မှာက job တခုရှိတယ် ၊ နံပါတ်က 1 ၊ အဲဒါကလည်း run နေတယ် ပြီးတော့ command က xlogo & ဖြစ်တယ်လို့ပြနေပါတယ် ။

Returning a Process to the Foreground

background ထဲက process တခုဟာ Ctrl+C အပါအဝင်သူ့ကိုရပ်ပစ်ဖို့ကြိုးစားတဲ့မည်သည့် keyboard input တွေ ကနေမဆိုကင်းလွတ်နေပါတယ် ။ process တခုကို foreground ကိုပြန်ရောက်စေချင်တယ်ဆိုရင် အောက်မှာပြထားတဲ့ ဥပမာ ထဲကလို fg command ကိုသုံးရမှာဖြစ်ပါတယ် ။

jok3r@lucy:~\$ jobs

[1]+ Running xlogo &

jok3r@lucy:~\$ fg %1

xlogo

fg command ရဲ့အနောက်မှာ (%) percent sign နဲ့ job number (jobspec လို့လည်းခေါ်တယ်) ကလှည့်ကွက်ကို လုပ်ပေးသွားတာပါပဲ ။ တကယ်လို့ကျွန်တော်တို့မှာ background job တခုပဲရှိမယ်ဆိုရင် jobspec ကမထည့်လည်းရပါတယ် ။ xlogo ကိုပိတ်ဖို့အတွက် Ctrl+C ကိုနှိပ်လိုက်ပါ ။

Stopping (Pausing) a Process

တခါတလေမှာကျွန်တော်တို့က process တခုကိုမပိတ်ပဲရပ်ထားချင်တာမျိုးရှိပါလိမ့်မယ် ။ ဒါကိုတခါတလေမှာ foreground process တခုကို background ကိုရွှေ့လိုက်ခြင်းဖြင့်လုပ်ဆောင်နိုင်ပါတယ် ။ foreground process တခုကိုရပ်စေ

ချင်ရင်တော့ Ctrl + Z ကိုနှိပ်လိုက်ပါ။ လုပ်ကြည့်ကြစို့။ command prompt မှာ xlogo လို့ရိုက်လိုက်ပါ။ Enter key ကိုနှိပ်ပါ။ ပြီးရင် Ctrl + Z ကိုနှိပ်လိုက်ပါ။ (Ubuntu 20.04 မှာတော့ Ctrl + Z ကိုမနှိပ်ခင် Alt+Tab နဲ့ဖြစ်ဖြစ် command prompt ဆီကို ပြန်သွားပြီးတော့မှ Ctrl + Z ကိုနှိပ်လို့ရပါတယ်။)

```
jok3r@lucy:~$ xlogo
^Z
[1]+  Stopped                  xlogo
jok3r@lucy:~$
```

xlogo ကိုရပ်ပြီးသွားတဲ့အခါကျွန်တော်တို့အနေနဲ့ program ရပ်မရပ်စမ်းစစ်ရန်အတွက် xlogo window ရဲ့ size ကို ချို့တာချဲ့တာလုပ်ကြည့်လိုက်ပါ။ ကျွန်တော်တို့အနေနဲ့ (xlogo အလုပ်မလုပ်တာ) သူကသေနေတာကိုတွေ့ရမှာဖြစ်ပါတယ်။ ကျွန်တော်တို့အနေနဲ့ fg command ကိုသုံးပြီးတော့ program ကို foreground ကိုပြန်ပို့တာ သို့မဟုတ် bg command ကိုသုံးပြီးတော့ program ကို background ကိုပြန်ပို့တာစတဲ့နှစ်မျိုးထဲကတမျိုးပြုလုပ်နိုင်ပါတယ်။

```
jok3r@lucy:~$ bg %1
[1]+ xlogo &
jok3r@lucy:~$
```

fg command မှာလိုပဲ တကယ်လို့အဲ့ဒီမှာ job ကတခုထဲရှိခဲ့မယ်ဆိုရင် jobspec က မထည့်လည်းရပါတယ်။ process တခုကို foreground ကနေ background ကိုပြောင်းတာဟာအသုံးဝင်ပါတယ်။ တကယ်လို့ကျွန်တော်တို့က graphical program တခုကို command ကနေ ဖွင့်လိုက်ပြီးတော့ trailing & ထည့်ပြီးသူ့ကို Background ကိုပြန်ပို့ဖို့မေ့နေတယ်ဆိုရင်ပေါ့။

ခင်ဗျားအနေနဲ့ဘာဖြစ်လို့ graphical program တခုကို command-line ကနေဖွင့်ချင်ရမှာလဲ ? အဲ့ဒီမှာအကြောင်းပြချက် ၂ ခုရှိပါတယ်။ ပထမတခုကတော့ ခင်ဗျား run ချင်နေတဲ့ program ဟာ window manager ရဲ့ menu ထဲမှာရှိမနေလို့ (ဥပမာ - xlogo လိုမျိုး)

ဒုတိယတခုကတော့ program တခုကို command-line ကနေဖွင့်ခြင်းအားဖြင့် ခင်ဗျားအနေနဲ့ တကယ်လို့ graphical အနေနဲ့ program ကိုဖွင့်တဲ့အခါမမြင်ရတဲ့ error message တွေကိုမြင်တွေ့နိုင်တာဖြစ်လို့ပါပဲ။ တခါတလေမှာ program တခုဟာ graphical ကနေဖွင့်တဲ့အခါ စတင်မပေးနိုင်တာမျိုးရှိပါတယ်။ အဲ့ဒီအစား သူ့ကို command-line ကနေ သွားဖွင့်တဲ့အခါ ကျွန်တော်တို့အနေနဲ့ ဘာပြဿနာဖြစ်နေသလဲဆိုတာကိုဖော်ပြပေးတဲ့ message ကိုမြင်တွေ့နိုင်မှာဖြစ်ပါတယ်။ တချို့ graphical program တွေမှာလည်းစိတ်ဝင်စားစရာကောင်းပြီးအသုံးဝင်တဲ့ command-line option တွေပါဝင်နေတတ်ပါတယ်။

Signals

kill command ကိုအသုံးပြုပြီးတော့ process တွေကို “kill” (terminate) ပြုလုပ်ပါတယ် ။ ဒါကကျွန်တော်တို့ကို ဆိုးရွားစွာပြုမူနေတဲ့ program တခုကို execution လုပ်နေရက သို့မဟုတ် သူ့ဘာသာသူရပ်တန့်ဖို့အတွက်ငြင်းဆန်နေတာ တွေကနေ ရပ်တန့်ခွင့်ပြုပါတယ် ။ ဒီမှာ ဥပမာ တခုပါ ။

```
jok3r@lucy:~$ xlogo &
[2] 94655
[1] Done          xlogo
jok3r@lucy:~$ kill 94655
jok3r@lucy:~$
```

ကျွန်တော်တို့အနေနဲ့ ပထမဆုံး xlogo ကို background မှာ ဖွင့်လိုက်ပါမယ် ။ shell က background process ရဲ့ jobspec နဲ့ PID ကိုထုတ်ပြပါလိမ့်မယ် ။ ပြီးရင်ကျွန်တော်တို့က kill command ကိုသုံးပြီးတော့ ကျွန်တော်တို့ ပိတ်ချင်တဲ့ process ရဲ့ PID ကိုအတိအကျညွှန်းပေးပါမယ် ။ ကျွန်တော်တို့အနေနဲ့ PID တခုအစား jobspec တခုနဲ့လည်း (%2) ဆိုပြီးတော့ အတိအကျညွှန်းပေးလို့လည်းရပါတယ် ။

ဒါတွေအကုန်လုံးကအရမ်းကိုတဲးတိုးဆန်ပေမယ့်လည်းသူ့မှာနောက်ထပ်ရှိပါသေးတယ် ။ kill command က process တွေကိုအတိအကျသွားပြီး kill ပေးတာမဟုတ်ပဲသူက signal ပဲပို့ပေးလိုက်တာပါ ။ signal ဆိုတာ operating system တွေက program တွေနဲ့ဆက်သွယ်ရာမှာအသုံးပြုတဲ့နည်းလမ်းတွေအများကြီးထဲကတခုပဲဖြစ်ပါတယ် ။ ကျွန်တော်တို့အနေနဲ့ signal တွေလက်တွေ့အလုပ်လုပ်ပုံကို Ctrl+C နဲ့ Ctrl+Z တို့ကိုအသုံးပြုစဉ်ကတွေ့မြင်ခဲ့ကြပြီးဖြစ်ပါတယ် ။ terminal ကအဲ့ဒီ keystroke တွေထဲကတခုခုကိုလက်ခံရရှိတဲ့အခါမှာ သူက foreground မှာရှိတဲ့ program ဆီကိုအဲ့ဒီ signal ကိုပို့ပေးလိုက်ပါတယ် ။ CTRL+C ကိစ္စမှာဆိုရင် INT (Interrupt) လို့ခေါ်တဲ့ signal ကိုပို့ပေးပြီးတော့ Ctrl+Z ကိစ္စမှာဆိုရင် TSTP (Terminal Stop) လို့ခေါ်တဲ့ signal ကိုပို့ပေးပါတယ် ။ Program တွေအနေနဲ့ကတော့ signal တွေကိုနားထောင်နေပြီးတော့သူတို့ရရှိတဲ့ အတိုင်းလုပ်ဆောင်ပေးပါတယ် ။ တကယ်တမ်း program တခုက signal ကိုနားထောင်ပြီးတော့အဲ့ဒီ signal ကခွင့်ပြုတဲ့ အတိုင်းလုပ်ဆောင်နိုင်တဲ့အထဲမှာ ရပ်တန့်ဖို့ signal ပေးတဲ့အခါ လုပ်လက်စတွေကို save ပေးတာမျိုးလည်းပါဝင်ပါတယ် ။

Sending Signals to Processes with kill

kill command ရဲ့အသုံးဆုံးဖြစ်တဲ့ syntax ပုံစံကတော့ဒီလိုမျိုးဖြစ်ပါတယ် ။

kill [-signal] PID ...

တကယ်လို့ command line ပေါ်မှာ signal ကိုတိတိကျကျမရည်ညွှန်းပေးထားခဲ့ဘူးဆိုရင်တော့ TERM (terminate) signal ကိုပဲ default အနေနဲ့ပို့ပေးပါတယ် ။ kill command ကို Table 10-4 မှာပြထားတဲ့ signal တွေ ပို့ဖို့အတွက်မကြာခဏသုံးလေ့ရှိပါတယ် ။

Table 10-4: Common Signals

Number	Name	Meaning
--------	------	---------

1	HUP	<p>Hang up (ဖုန်းချတာ) ။ ဒါကတော့ terminal တွေက remote computer တွေဆီကို ဖုန်းလိုင်းတွေ modem တွေနဲ့ချိတ်ဆက်အသုံးပြုခဲ့ကြတဲ့ကောင်းမွန်တဲ့ခေတ်ဟောင်း ရဲ့လက်ကျန်တွေထဲကတစ်ခုပါပဲ ။ ဒီ signal တွေကို controlling terminal တွေက Hang up ဖုန်းချသွားပြီဖြစ်ကြောင်း program ကိုပြောတဲ့အခါမှာသုံးပါတယ် ။ ဒီ signal ရဲ့ effect မျိုးကို terminal session တစ်ခုကိုပိတ်ချလိုက်ခြင်းဖြင့်လက်တွေ့လုပ်ပြလို့ရပါတယ် ။ terminal ပေါ်မှာ run နေတဲ့ foreground program ဟာဒီ signal အပို့ခံရပြီးတော့သူကပိတ်ချလိုက်မှာဖြစ်ပါတယ် ။</p> <p>ဒီ signal ကို daemon program များစွာက reinitialization တစ်ခုဖြစ်စေဖို့အတွက် လုပ်တဲ့နေရာမှာလည်းသုံးကြပါတယ် ။ ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ daemon က ဒီ signal ကိုပေးပို့ခြင်းခံရတဲ့အခါသူက restart လုပ်ပြီးတော့သူ့ရဲ့ configuration file ကိုပြန်ဖတ်ပေးမှာဖြစ်ပါတယ် ။ Apache web server ဟာ HUP signal တွေကိုဒီနည်းလမ်းအတိုင်း အသုံးပြုတဲ့ daemon တွေထဲကဥပမာတစ်ခုဖြစ်ပါတယ် ။</p>
2	INT	Interrupt ၊ terminal ကနေ Ctrl+C key ပို့လိုက်တဲ့အခါလုပ်ဆောင်ပေးတဲ့ function နဲ့အတူတူပဲလုပ်ဆောင်ပေးပါတယ် ။ ဒါကသာမန်အားဖြင့်တော့ program တစ်ခုကိုပိတ်ချပေးပါတယ် ။
9	KILL	kill ၊ ဒီ signal ကအရမ်းကိုထူးခြားပါတယ် ။ program တွေအနေနဲ့သူတို့ဆီကိုပို့လိုက်တဲ့ signal တွေကို အကုန်လုံးကို ignore လုပ်ပစ် (လျစ်လျူရှု) လိုက်တဲ့နည်းလမ်းအပါအဝင် မတူညီတဲ့နည်းလမ်းတွေနဲ့ကိုင်တွယ်ပါတယ် ။ kill signal ဟာဘယ်တုန်းကမှ target program ဆီကိုသေချာပို့လိုက်တာမျိုးမဟုတ်ပါဘူး ။ အဲဒီအစား kernel က process ကိုချက်ချင်းရပ်ပစ်လိုက်တာဖြစ်ပါတယ် ။ process တစ်ခုကိုဒီလိုပုံစံမျိုးနဲ့ ပိတ်ချပစ်လိုက်တဲ့အခါမှာ သူ့ကိုယ်တိုင်နဲ့သူ့ရဲ့မ save ရသေးတဲ့အလုပ်တွေအတွက် clean up လုပ်ဖို့အခွင့်အရေးမပေးတော့ပါဘူး ။ ဒီအကြောင်းကြောင့် အခြား termination signal တွေနဲ့လုပ်လို့မရတော့မှာသာ KILL signal ကိုနောက်ဆုံးမတတ်သာလို့သုံးရတဲ့အနေအထားအဖြစ်သုံးသင့်ပါတယ် ။
15	TERM	Terminate ၊ ဒါက kill command ကပို့တဲ့ default signal ဖြစ်ပါတယ် ။ တကယ်လို့ program တစ်ခုကို signal လက်ခံနိုင်လောက်တဲ့အထိအသက်ရှိနေသေးမယ်ဆိုရင်သူကပိတ်ချပေးမှာဖြစ်ပါတယ် ။
18	CONT	Continue ၊ ဒါက STOP signal တစ်ခုပေးပြီးတဲ့နောက်မှာ process ကိုပြန်ခေါ်ပေးမှာဖြစ်ပါတယ် ။
19	STOP	Stop ၊ ဒီ signal က process တစ်ခုကိုမပိတ်ချပဲခဏရပ်တန့်ပေးထားမှာဖြစ်ပါတယ် ။ KILL signal လိုပဲသူကလည်း target process ဆီကိုမပို့ပါဘူး ။ ဒါကြောင့်သူ့ကို ignore လျစ်လျူရှုလို့မရနိုင်ပါဘူး ။

kill command ကိုစမ်းကြည့်ကြစို့ ။

[1] 97344

jok3r@lucy:~\$ kill -1 97344

[1]+ Interrupt xlogo

jok3r@lucy:~\$

အခု ဥပမာထဲမှာ ကျွန်တော်တို့က xlogo program ကို background မှာစတင်လိုက်ပြီးတော့ Kill နဲ့အတူ HUP signal ကိုပေးပို့လိုက်ပါတယ်။ xlogo program ပိတ်သွားပြီးတော့ background process က hangup signal တခုလက်ခံရရှိကြောင်း shell ကညွှန်ပြနေပါတယ်။ ခင်ဗျားအနေနဲ့ message ကိုမြင်ရဖို့အတွက် ENTER key ကို အချက်အနည်းငယ်နှိပ်ပေးဖို့လိုကောင်းလိုပါလိမ့်မယ်။ (Ubuntu 20.04 မှာဆိုရင် enter ၂ ချက်နှိပ်ရပါမယ်) တခုမှတ်ထားရမှာက signal တွေကို နံပါတ် သို့မဟုတ် နာမည် ၊ ဂုဏ်သတ္တိတခုနဲ့ SIG ဆိုတဲ့စာနဲ့ name prefixed လုပ်တာ အပါအဝင် အတိအကျရည်ညွှန်းပေးရမှာဖြစ်ပါတယ်။

jok3r@lucy:~\$ xlogo &

[1] 97665

jok3r@lucy:~\$ kill -INT 97665

jok3r@lucy:~\$

[1]+ Interrupt xlogo

jok3r@lucy:~\$ xlogo &

[1] 97667

jok3r@lucy:~\$ kill -SIGINT 97667

jok3r@lucy:~\$

[1]+ Interrupt xlogo

jok3r@lucy:~\$

ဒီအပေါ်ကဥပမာကိုပဲတခြား signal တွေနဲ့ထပ်လုပ်ပြီးစမ်းကြည့်ပါ။ တခုမှတ်ထားရမှာကခင်ဗျားအနေနဲ့ PID တွေရဲ့နေရာမှာ jobspec တွေလည်းထည့်လို့ရတယ်ဆိုတာပါပဲ။

process တွေဟာ file တွေလိုပဲပိုင်ရှင် owner တွေရှိပြီးတော့ ခင်ဗျားအနေနဲ့ process ရဲ့ owner (သို့မဟုတ် superuser) ဖြစ်မှသာ kill နဲ့ signal ပို့ခွင့်ရှိမှာဖြစ်ပါတယ်။

နောက်တိုးအနေနဲ့ Table 10-4 မှာ list ပြထားတဲ့ signal တွေဟာ kill နဲ့မကြာခဏတွဲသုံးလေ့ရှိပြီးတော့ အခြား signal တွေကိုတော့ system ကမကြာခဏသုံးပါတယ်။ Table 10-5 မှာတော့အခြားအသုံးများတဲ့ signal တွေကို list လုပ်ပြထားပါတယ်။

Table 10-5: Other Common Signals

Number	Name	Meaning
3	QUIT	Quit ထွက်တာပါ။
11	SEGV	Segmentation violation ၊ တကယ်လို့ program က တရားမဝင် memory ကိုသုံးစွဲနေတယ်ဆိုရင်ဒီ signal ကိုပို့ပါတယ်။ ။ ဒါပဲလေ။ ။ သူကသူ့ကိုခွင့်မပြုထားတဲ့နေရာမှာ write လုပ်ဖို့ကြိုးစားနေတာကိုး။
20	TSTP	Terminal stop ၊ ဒီ signal ကို terminal က Ctrl+Z နှိပ်တဲ့အခါမှာပို့ပါတယ်။ ။ STOP signal နဲ့မတူတာက TSTP signal ကို program ကလက်ခံရတာဖြစ်ပြီး program က သူ့ကို ignore လုပ်ဖို့ရွေးချယ်ခွင့်ရှိပါတယ်။ ။
28	WINCH	Window change ၊ ဒါက window ရဲ့ size change သွားတဲ့အခါမှာ system ကပို့တဲ့ signal တခုဖြစ်ပါတယ်။ ။ top နဲ့ less လိုတခြား program တွေမှာဆိုရင်တော့သူတို့ကိုယ်သူတို့ window dimension အသစ်ထဲမှာဝင်ဆန့်အောင်ပြန်ပြင်ဆွဲခြင်းအားဖြင့် ဒီ signal ကိုတုန့်ပြန်ကြပါတယ်။ ။

စပ်စုချင်တဲ့သူတွေ (အခုထက်ပိုသိချင်တဲ့သူတွေ)အတွက်အောက်ပါ command နဲ့ signal list အပြည့်အစုံကိုကြည့်နိုင်ပါတယ်။ ။

```
jok3r@lucy:~$ kill -l
```

- | | | | | |
|-----------------|-----------------|-----------------|-----------------|-----------------|
| 1) SIGHUP | 2) SIGINT | 3) SIGQUIT | 4) SIGILL | 5) SIGTRAP |
| 6) SIGABRT | 7) SIGBUS | 8) SIGFPE | 9) SIGKILL | |
| 10) SIGUSR1 | | | | |
| 11) SIGSEGV | 12) SIGUSR2 | 13) SIGPIPE | 14) SIGALRM | |
| 15) SIGTERM | | | | |
| 16) SIGSTKFLT | 17) SIGCHLD | 18) SIGCONT | 19) SIGSTOP | |
| 20) SIGTSTP | | | | |
| 21) SIGTTIN | 22) SIGTTOU | 23) SIGURG | 24) SIGXCPU | |
| 25) SIGXFSZ | | | | |
| 26) SIGVTALRM | 27) SIGPROF | 28) SIGWINCH | 29) SIGIO | 30) SIGPWR |
| 31) SIGSYS | 34) SIGRTMIN | 35) SIGRTMIN+1 | 36) SIGRTMIN+2 | 37) SIGRTMIN+3 |
| 38) SIGRTMIN+4 | 39) SIGRTMIN+5 | 40) SIGRTMIN+6 | 41) SIGRTMIN+7 | 42) SIGRTMIN+8 |
| 43) SIGRTMIN+9 | 44) SIGRTMIN+10 | 45) SIGRTMIN+11 | 46) SIGRTMIN+12 | 47) SIGRTMIN+13 |
| 48) SIGRTMIN+14 | 49) SIGRTMIN+15 | 50) SIGRTMAX-14 | 51) SIGRTMAX-13 | 52) SIGRTMAX-12 |
| 53) SIGRTMAX-11 | 54) SIGRTMAX-10 | 55) SIGRTMAX-9 | 56) SIGRTMAX-8 | 57) SIGRTMAX-7 |

58) SIGRTMAX-6 59) SIGRTMAX-5 60) SIGRTMAX-4 61) SIGRTMAX-3 62) SIGRTMAX-2
63) SIGRTMAX-1 64) SIGRTMAX
jok3r@lucy:~\$

Sending Signals to Multiple Processes with killall

တိကျတဲ့ username သို့မဟုတ် program တခုရဲ့ multi process (တခုထက်ပိုတဲ့ process တွေ) ကို လည်း killall command ကိုအသုံးပြုပြီးတော့ signal တွေပို့လိုရပါတယ်။ syntax ကတော့ဒီမှာပါ။

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

လက်တွေ့လုပ်ပြဖို့အတွက်ကျွန်တော်တို့က xlogo program တွေအများကြီးဖွင့်ထားပြီးတော့ပြန်ပိတ်ပြမှာ ဖြစ်ပါတယ်။

```
jok3r@lucy:~$ xlogo &
[1] 98737
jok3r@lucy:~$ xlogo &
[2] 98741
jok3r@lucy:~$ killall xlogo
jok3r@lucy:~$
[1]- Terminated          xlogo
[2]+ Terminated          xlogo
jok3r@lucy:~$
```

တခုမှတ်ထားရမှာက kill နဲ့ဆိုရင် ခင်ဗျားနဲ့မသက်ဆိုင်တဲ့ process တွေဆီကို signal ပို့ဖို့အတွက် superuser privileges မရှိမဖြစ်လိုအပ်မှာဖြစ်ပါတယ်။

More Process-Related Commands

monitoring processe တွေဟာအရေးကြီးတဲ့ system administration task တွေဖြစ်တာကြောင့် သူ့အတွက် command တွေအများကြီးရှိနေပါတယ်။ Table 10-6 မှာစမ်းသပ်ကြည့်ဖို့အချို့ကို list လုပ်ပေးထားပါတယ်။

Table 10-6: Other Process-Related Commands

Command	Description
ps tree	process list တစ်ခုကိုသစ်ပင်ပုံစံ pattern နဲ့စီစဉ်ထားပြီးတော့ process တွေကြားမှာ parent/child ဆက်နွယ်မှု ကိုပြနေတဲ့အနေနဲ့ output ထုတ်ပေးပါတယ် ။
vmstat	memory, swap နဲ့ disk I/O အပါအဝင် system resource အသုံးပြုမှုရဲ့ snapshot တစ်ခုကို output ထုတ်ပေးပါတယ် ။ ဆက်တိုက်ပြနေတာကိုမြင်ရဖို့အတွက် command ရဲ့အနောက်မှာ time delay (in seconds) ထည့်ပေးခြင်းဖြင့် update ကြည့်နိုင်ပါတယ် ။ (ဥပမာ - vmstat 5) output ကိုရပ်ချင်ရင် CTRL -C နှိပ်ပါ ။
xload	အချိန်နဲ့အမျှ system load ကို graph တစ်ခုအနေနဲ့ဆွဲပြတဲ့ graphical program တစ်ခုဖြစ်ပါတယ် ။
tload	xload program နဲ့ဆင်တူပါတယ် ။ ဒါပေမယ့် graph ကို terminal ထဲမှာဆွဲပြပါတယ် ။ output ကို ရပ်ချင်ရင် CTRL -C နှိပ်ပါ ။

PART 2

CONFIGURATION AND THE ENVIRONMENT

11

THE ENVIRONMENT

ကျွန်တော်တို့စောစောပိုင်းကဆွေးနွေးခဲ့သလိုပဲ ကျွန်တော်တို့ရဲ့ shell session က environment ကိုခေါ်ပေးနေတဲ့အချိန်မှာ shell က information ရဲ့ body ကို maintain လုပ်ပေးနေပါတယ်။ environment ထဲမှာ သိမ်းထားတဲ့ data တွေကို program တွေကအသုံးပြုပြီးတော့ ကျွန်တော်တို့ရဲ့ configuration အတွက်အမှန်တရားကိုဆုံးဖြတ်ပေးပါတယ်။ program အများစုက configuration file တွေကိုအသုံးပြုပြီးတော့ program setting တွေကိုသိမ်းထားနေစဉ်မှာ အချို့ program တွေကလည်းပဲသူတို့ရဲ့ behavior ကိုချိန်ညှိဖို့အတွက် environment ထဲမှာသိမ်းထားတဲ့ value တွေကိုလိုက်ရှာနေပါတယ်။ ဒါကိုသိရင် ၊ ကျွန်တော်တို့က environment ကိုအသုံးပြုပြီးတော့ shell experience ကို customize လုပ်လို့ရပါတယ်။

အခု Chapter မှာကျွန်တော်တို့ကအောက်ပါ command တွေအလုပ်လုပ်မှာဖြစ်ပါတယ် ။

- **printenv** — environment ရဲ့အစိတ်အပိုင်း သို့မဟုတ်အကုန်လုံးကို Print ထုတ်ပြမယ် ။
- **set** — shell option တွေကို set လုပ်မယ် ။
- **export** — subsequently executed program တွေဆီကို environment ကို Export ထုတ်မယ် ။
- **alias** — command တခုအတွက် alias ဖန်တီးမယ် ။

What Is Stored in the Environment?

shell က environment ထဲမှာအခြေခံ data type ၂ မျိုးကိုသိမ်းထားလေ့ရှိသည့်တိုင်အောင် bash နဲ့ဆိုရင် အဲဒီ type တွေဟာမခွဲခြားနိုင်မှုကြီးကြီးမားမားရှိပါတယ် ။ အဲဒါတွေကတော့ environment variable တွေနဲ့ shell variable တွေပဲဖြစ်ပါတယ် ။ Shell variable တွေဟာ data အပိုင်းအစလေးတွေဖြစ်ပြီးတော့သူတို့ကိုအဲဒီမှာ Bash ကသွားထားထားတာဖြစ်ပြီးတော့ environment variable တွေကတော့အခြေခံအားဖြင့်ကျန်တဲ့အရာတွေအကုန်လုံး ပဲဖြစ်ပါတယ် ။ variable တွေမှာထပ်တိုးအနေနဲ့ ၊ shell က program နဲ့ဆိုင်တဲ့ data တွေကိုလည်းသိုလှောင်ထား ပေးပါသေးတယ် ။ အတိအကျပြောရမယ်ဆိုရင်တော့ alias တွေနဲ့ shell function တွေပါပဲ ။ ကျွန်တော်တို့ Chapter 5 တုန်းက alias တွေအကြောင်းလေ့လာခဲ့ကြပြီးတော့ shell function တွေ (shell scripting နဲ့ဆိုင်တဲ့အရာတွေ) ကိုတော့ Part 4 ကျရင်ဆွေးနွေးပေးသွားမှာဖြစ်ပါတယ် ။

Examining the Environment

environment ထဲမှာဘာတွေသိုလှောင်သိမ်းဆည်းထားသလဲဆိုတာကိုကြည့်ဖို့အတွက်ကျွန်တော်တို့အနေ နဲ့(Os အထဲမှာ) set လုပ်ပေးထားတဲ့ bash သို့မဟုတ် printenv program ၂ ခုထဲကတခုကိုအသုံးပြုနိုင်ပါတယ် ။ set command တွေဟာ shell နဲ့ environment variable ၂ ခုစလုံးကိုဖော်ပြပေးမှာဖြစ်သော်လည်း printenv ကတော့ နောက်ပိုင်းကဟာတွေကိုပဲဖော်ပြပေးမှာဖြစ်ပါတယ် ။ environment content တွေရဲ့ list ဟာအတော်လေး ရှည်လျားတာကြောင့် command ၂ ခုလုံးရဲ့ output တွေကို less နဲ့ pipe တွဲပြီးထုတ်တာအကောင်းဆုံးဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ printenv | less
```

အဲဒီလိုလုပ်ခြင်းအားဖြင့်ကျွန်တော်တို့မှာဒီလိုဟာနဲ့တူတာတခုရရမှာဖြစ်ပါတယ် ။

```
SHELL=/bin/bash
```

```
SESSION_MANAGER=local/lucy:@/tmp/.ICE-unix/1537,unix/lucy:/tmp/.ICE-unix/1537
```

```
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu-wayland:/etc/xdg
SSH_AGENT_LAUNCHER=openssh
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
LC_ADDRESS=en_US.UTF-8
GNOME_SHELL_SESSION_MODE=ubuntu
LC_NAME=en_US.UTF-8
SSH_AUTH_SOCK=/run/user/1000/openssh_agent
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu-wayland
LC_MONETARY=en_US.UTF-8
GTK_MODULES=gail:atk-bridge
PWD=/home/jok3r
LOGNAME=jok3r
XDG_SESSION_DESKTOP=ubuntu-wayland
QSYS_ROOTDIR=/home/jok3r/intelFPGA_lite/20.1/quartus/sopc_builder/bin
XDG_SESSION_TYPE=wayland
XAUTHORITY=/run/user/1000/.mutter-Xwaylandauth.JFHNN1
:
```

ကျွန်တော်တို့မြင်နေရတာကတော့ environment variable နဲ့သူ့ရဲ့ value တွေပါဝင်တဲ့ list တခုပဲဖြစ်ပါတယ် ။ ဥပမာ - ကျွန်တော်တို့က jok3r ဆိုတဲ့ value ပါဝင်နေတဲ့ USER ဆိုတဲ့ variable တခုကိုမြင်နေရပါတယ် ။ printenv command ကလည်းပဲ သတ်မှတ်ထားတဲ့ variable တခုရဲ့ value ကို list ထုတ်ပေးနိုင်ပါတယ် ။

```
jok3r@lucy:~$ printenv USER
jok3r
jok3r@lucy:~$
```

set command ကို option တွေသို့မဟုတ် argument တွေမပါပဲအသုံးပြုတဲ့အခါ သူက shell နဲ့ environment variable တွေအတူ define လုပ်ထားတဲ့ shell function တွေကိုပါဖော်ပြပေးမှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ set | less
```

printenv နဲ့မတူတာက ၊ သူ့ရဲ့ output ကို (အင်္ဂလိပ်အက္ခရာစဉ်အတိုင်း) ကောင်းမွန်သေသပ်စွာစီထားပေးခြင်းပဲဖြစ်ပါတယ် ။ အဲဒါက single variable တခုထဲက content တွေကို echo command ကိုအသုံးပြုပြီးတော့ အောက်ပါအတိုင်း ကြည့်လို့လည်းရပါတယ် ။

```
jok3r@lucy:~$ echo $HOME
```

set သို့မဟုတ် printenv မဟုတ်ပဲ display ဖော်ပြပေးတဲ့ environment ရဲ့ element တွေထဲကတခုကတော့ alias ပဲဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ alias
alias alert='notify-send --urgency=low -i "${[ $? = 0 ]} && echo terminal || echo error)" "$(history|tail -n1|sed -e
\'\'s/^s*[0-9]+\s*//;s/[:;&]\s*alert$/\'\'")\'\'
alias egrep='egrep --color=auto'
alias fgrep='fgrep --color=auto'
alias grep='grep --color=auto'
alias l='ls -CF'
alias la='ls -A'
alias ll='ls -alF'
alias ls='ls --color=auto'
jok3r@lucy:~$
```

Some Interesting Variables

environment မှာ variable အများအပြားပါဝင်နေသည့်တိုင်အောင်ခင်ဗျားရဲ့ environment ဟာအခုပြထားတဲ့ဟာနဲ့ကွဲပြားချင်ကွဲပြားနေနိုင်သော်လည်း ခင်ဗျားအနေနဲ့ခင်ဗျားရဲ့ environment ထဲမှာ Table 11-1 မှာပြထားတဲ့ variable တွေနဲ့တွေ့ရဖို့များပါတယ် ။

Table 11-1: Environment Variables

Variable	Contents
DISPLAY	တကယ်လို့ခင်ဗျားက Graphical display တခုကို run နေတယ်ဆိုရင် ဒါကခင်ဗျားရဲ့ display နာမည်ပဲဖြစ်ပါတယ် ။ ပုံမှန်အားဖြင့်တော့ :0 (zero) ပါ ။ ဆိုလိုတာက xserver က generate လုပ်ပေးလိုက်တဲ့ပထမဆုံး display ဖြစ်ပါတယ် ။

EDITOR	text editing လုပ်တဲ့နေရာမှာသုံးတဲ့ program ရဲ့နာမည်ဖြစ်ပါတယ်။
SHELL	ခင်ဗျားရဲ့ shell program နာမည်ဖြစ်ပါတယ်။
HOME	ခင်ဗျားရဲ့ home directory ရဲ့ pathname ဖြစ်ပါတယ်။
LANG	ခင်ဗျားရဲ့ language ရဲ့ collation order နဲ့ character set တို့ကိုသတ်မှတ်ပေးတာဖြစ်ပါတယ်။
OLD_PWD	နောက်ဆုံးအလုပ်လုပ်ခဲ့တဲ့ working directory ဖြစ်ပါတယ်။
PAGER	output တွေကို paging လုပ်တဲ့နေရာမှာသုံးတဲ့ program ရဲ့နာမည်ဖြစ်ပါတယ်။ ဒါကိုတခါတလေမှာ /usr/bin/less ထဲမှာ set လုပ်ထားလေ့ရှိပါတယ်။
PATH	ခင်ဗျားအနေနဲ့ executable program တခုရဲ့နာမည်ကိုရိုက်ထည့်လိုက်တဲ့အခါမှာသူက colon (:) တွေနဲ့ခြားထားတဲ့ directory list ထဲမှာသွားရှာပေးပါတယ်။
PS1	Prompt String 1 ဖြစ်ပါတယ်။ အဲဒါကခင်ဗျားရဲ့ shell prompt ကိုသတ်မှတ်ပေးပါတယ်။ ကျွန်တော်တို့နောက်ပိုင်းကျရင်တွေ့မှာဖြစ်တဲ့အတိုင်းအဲဒါက တော်တော်လေးကို customize လုပ်လို့ရပါတယ်။
PWD	လက်ရှိရောက်နေတဲ့နေရာ current working directory ဖြစ်ပါတယ်။
TERM	ခင်ဗျားရဲ့ terminal type ရဲ့နာမည်ဖြစ်ပါတယ်။ Unix-like system တွေမှာ terminal protocol တွေအများအပြားကို support ပေးပါတယ်။ ဒီ variable တွေကခင်ဗျားရဲ့ terminal emulator နဲ့အသုံးပြုဖို့အတွက် protocol ကို set လုပ်ပေးပါတယ်။
TZ	ခင်ဗျားရဲ့ time zone ကိုအတိအကျသတ်မှတ်ပေးပါတယ်။ Unix-like system တွေအများစုမှာ computer ရဲ့ internal clock ကို Coordinated Universal Time (UTC) နဲ့သတ်မှတ်ထိန်းသိမ်းထားပြီးတော့ local time ကိုဖော်ပြဖို့အတွက်ဒီ variable ကသတ်မှတ်ပေးတဲ့ offset နဲ့ပြုလုပ်ပါတယ်။
USER	ခင်ဗျားရဲ့ username ပဲဖြစ်ပါတယ်။

တချို့ value တွေပျောက်နေရင်စိတ်မပူပါနဲ့။ အဲ့တွေက Linux distribution အပေါ်မူတည်ပြီးကွဲပြားတတ်ပါတယ်။

How Is the Environment Established?

ကျွန်တော်တို့က system ပေါ်ကို log on ဝင်လိုက်တဲ့အခါမှာ bash program က startup files လို့ခေါ်တဲ့ configuration script တွေအတွဲလိုက်ကိုစတင်ပြီးတော့ဖတ်ပေးပါတယ်။ အဲဒါကလည်း user တွေအကုန်လုံး share လုပ်သုံးစွဲဖို့အတွက် default environment ကိုသတ်မှတ်ပေးပါတယ်။ ဒါက ကျွန်တော်တို့ရဲ့ home directory ထဲမှာ နောက်ထပ် startup file တွေနောက်ကထပ်လိုက်လာစေပြီးတော့ကျွန်တော်တို့ရဲ့ personal environment ကို သတ်မှတ်ပေးစေပါတယ်။ တိကျတဲ့ sequence တွေဟာ စတင်လိုက်တဲ့ shell session အမျိုးအစားပေါ်မူတည်နေပါတယ်။

Login and Non-login Shells

shell session အမျိုးအစား ၂ မျိုးရှိပါတယ် ။ တခုက login shell session ဖြစ်ပြီးနောက်တခုက non-login shell session ဖြစ်ပါတယ် ။

login shell session ဆိုတာကျွန်တော်တို့ကို prompt ကနေ username နဲ့ password တောင်းတဲ့ဟာတခုကိုပြောတာပါ ။ ဥပမာ - ကျွန်တော်တို့က virtual console session တခုကိုစတင်လိုက်တဲ့အခါမှာပေါ့ ။ non-login shell session တခုကတော့သာမန်အားဖြင့်ကျွန်တော်တို့က terminal session တခုကို GUI ကနေဖွင့်လိုက်တဲ့အခါမှာဖြစ်ပေါ်လာတဲ့ (shell session) ကိုပြောတာဖြစ်ပါတယ် ။

Login shell တွေဟာ တခု သို့မဟုတ် တခုထက်ပိုတဲ့ startup file တွေကို Table 11-2 မှာပြထားတဲ့အတိုင်းဖတ်ပေးပါတယ် ။

Table 11-2: Startup Files for Login Shell Sessions

File	Contents
/etc/profile	user တွေအကုန်လုံးကို apply လုပ်တဲ့ global configuration script တခုဖြစ်ပါတယ် ။
~/.bash_profile	user တယောက်ရဲ့ personal startup file ဖြစ်ပါတယ် ။ global configuration script ထဲက setting မှာ extend သို့မဟုတ် override လုပ်တဲ့အခါသုံးနိုင်ပါတယ် ။
~/.bash_login	တကယ်လို့ ~/.bash_profile မှာ ဆိုတဲ့ file ကိုရှာမတွေ့ရင် bash ကဒီ file ကိုဖတ်ဖို့ကြိုးစားပါလိမ့်မယ်
~/.profile	တကယ်လို့ ~/.bash_profile သို့မဟုတ် ~/.bash_login ၂ file ထဲကတခုခုကိုရှာတွေ့တဲ့အခါ bash ကဒီ file ကိုဖတ်ဖို့ကြိုးစားပါလိမ့်မယ် ။ ဒါက Ubuntu လိုမျိုး Debian base distribution တွေမှာတော့ default အနေနဲ့ပါပါတယ် ။

Non-login shell session တွေကတော့ startup file တွေကို Table 11-3 မှာပြထားတဲ့အတိုင်းဖတ်ပေးပါတယ် ။

Table 11-3: Startup Files for Non-Login Shell Sessions

File	Contents
/etc/bash.bashrc	user တွေအကုန်လုံးကို apply လုပ်တဲ့ global configuration script တခုဖြစ်ပါတယ် ။
~/.bashrc	user တယောက်ရဲ့ personal startup file ဖြစ်ပါတယ် ။ global configuration script ထဲက setting မှာ extend သို့မဟုတ် override လုပ်တဲ့အခါသုံးနိုင်ပါတယ် ။

အပေါ်က startup file တွေကိုဖတ်ဖို့အတွက်နောက်တခုက non-login shells တွေဟာသူတို့ရဲ့ parent process တွေဆီကနေ environment ကိုအမွေဆက်ခံထားလေ့ရှိပါတယ် ။ ပုံမှန်အားဖြင့်တော့ log in shell တခုပေါ့ ။

ခင်ဗျားရဲ့ system မှာ ဒီထဲကဘယ် startup file တွေပါဝင်နေသလဲဆိုတာကိုတချက်လောက်ကြည့်လိုက်ပါ ။ တခုမှတ်ထားပါ ။ အပေါ်မှာ list ပြထားတဲ့ filename တွေအများစုဟာ period (.) နဲ့စတင်ထားတာမို့ (ဆိုလိုတာကသူတို့ဟာ

hidden ဖွက်ထားတဲ့ file တွေဖြစ်တယ်) ခင်ဗျားအနေနဲ့ ls command ကိုသုံးတဲ့အခါ (အဲ့ file တွေကိုမြင်ရအောင်) -a option ကိုထည့်ပေးရမှာဖြစ်ပါတယ် ။

ရိုးရိုးသာမန် user အနေနဲ့ကြည့်မယ်ဆိုရင် ~/.bashrc file ဟာအမြဲတမ်းနီးပါး read လုပ်ခံရတဲ့အတွက် အရေးအကြီးဆုံး startup file တခုဖြစ်ပါတယ် ။ Non-login shell တွေက default အနေနဲ့သူ့ကိုသွားဖတ်ပေးပြီးတော့ login shell တွေရဲ့ startup file တွေဟာ ~/.bashrc file ကိုဖတ်ဖို့အတွက်လည်းပဲ အဲ့ဒီလိုနည်းလမ်းတခုနဲ့ရေးသားထားလေ့ရှိပါတယ် ။

What's in a Startup File?

တကယ်လို့ ကျွန်တော်တို့က သာမန် .bash_profile ကိုတချက်ကြည့်လိုက်မယ်ဆိုရင်သူကဒီလိုမျိုးနဲ့တူတာတခုခု ကိုပြပေးမှာဖြစ်ပါတယ် ။

```
jok3r@lucy:~$ /etc/profile
```

```
# /etc/profile: system-wide .profile file for the Bourne shell (sh(1))
```

```
# and Bourne compatible shells (bash(1), ksh(1), ash(1), ...).
```

```
if [ "${PS1-}" ]; then
```

```
    if [ "${BASH-}" ] && [ "$BASH" != "/bin/sh" ]; then
```

```
        # The file bash.bashrc already sets the default PS1.
```

```
        # PS1='\h:\w\$ '
```

```
        if [ -f /etc/bash.bashrc ]; then
```

```
            . /etc/bash.bashrc
```

```
        fi
```

```
    else
```

```
        if [ "`id -u`" -eq 0 ]; then
```

```
            PS1='# '
```

```
        else
```

```
            PS1='$ '
```

```
        fi
```

```
:
```

နဲ့စတဲ့စာကြောင်း (line) တွေဟာ comment တွေဖြစ်ပြီးတော့ shell ကမဖတ်ပေးပါဘူး ။ အဲ့ဒါတွေကိုအဲ့ဒီမှာ လူသားတွေဖတ်နိုင်ဖို့ထားထားပေးတာပါ ။ ပထမဦးဆုံးစိတ်ဝင်စားစရာကောင်းစေတာက အောက်ဖော်ပြပါ code တွေပါတဲ့ စာကြောင်း (line) ပဲဖြစ်ပါတယ် ။

```
if [ -f /etc/bash.bashrc ]; then
    . /etc/bash.bashrc
fi
```

ဒါကို if compound command လို့ခေါ်ပြီးတော့ အဲဒါကိုကျွန်တော်တို့က Part 4 က shell scripting ကျရင်လေ့လာ ကြမှာဖြစ်ပေမယ့်အခုကတော့သူ့ကိုဘာသာပြန်ပြမှာဖြစ်ပါတယ်။ ။

```
တကယ်လို့ [ -f /etc/bash.bashrc ]; ဆိုတဲ့ file ရှိခဲ့မယ်ဆိုရင်
. /etc/bash.bashrc ဆိုတဲ့ file ကိုသွားဖတ်ပေးမယ်။ ။
```

ကျွန်တော်တို့အနေနဲ့ဒီလို code အပိုင်းအစကလေးဟာ login shell တခုက .bashrc ထဲက content တွေကိုဘယ်လို ရယူသလဲဆိုတာကိုမြင်တွေ့နိုင်ပါတယ်။ ။ ကျွန်တော်တို့ရဲ့ startup file နဲ့ပတ်သက်တာနောက်တခုကတော့ PATH variable ဝဲ ဖြစ်ပါတယ်။ ။

ကျွန်တော်တို့က command line ပေါ်မှာ command တွေရှိနေထည့်လိုက်တဲ့အခါမှာ command တွေကို shell က ဘယ်မှာသွားရှာရမယ်ဆိုတာဘယ်လိုသိသလဲ ဆိုတာကိုတွေးကြည့်ဖူးလား ? ဥပမာ - ကျွန်တော်တို့က ls ကိုရှိနေထည့်လိုက်တဲ့ အခါမှာ shell က computer တခုလုံးမှာ /bin/ls (ls command ရဲ့ path name အပြည့်အစုံ) ကိုလိုက်ရှာပေးတာမဟုတ်ပဲ အဲ ဒီအစားသူက PATH variable ထဲက directory တွေရဲ့ list မှာသွားရှာပေးတာဖြစ်ပါတယ်။ ။

PATH variable ကတခါတလေ (အမြဲတမ်းမဟုတ်ပါ ၊ distribution အပေါ်မူတည်ပါတယ်) /etc/profile startup file ရဲ့ set လုပ်တာကိုခံရပြီးတော့ အောက်က code နဲ့ဆိုရင်

```
PATH=$PATH:$HOME/bin
```

PATH ဟာ \$HOME/bin ဆိုတဲ့ directory ကို list ရဲ့အဆုံးမှာသွားပေါင်းထည့်ပေးပါတယ်။ ။ ဒါက Chapter 7 မှာ တုန်းကကိုင်တွယ်ခဲ့တဲ့ parameter expansion ရဲ့ဥပမာတခုဖြစ်ပါတယ်။ ။ ဒါဘယ်လိုအလုပ်လုပ်သလဲဆိုတာကိုလက်တွေ့ပြဖို့ အတွက် အောက်ကဟာကိုစမ်းသပ်ကြည့်ပါ။ ။ (space မခြားပဲရိုက်ပါ။)

```
jok3r@lucy:~$ foo="This is some"
jok3r@lucy:~$ echo $foo
This is some
jok3r@lucy:~$ foo=$foo"Text."
jok3r@lucy:~$ echo $foo
```

This is someText.

jok3r@lucy:~\$

ဒီနည်းစနစ်ကိုအသုံးပြုခြင်းအားဖြင့်ကျွန်တော်တို့က variable ထဲက content တွေရဲ့အဆုံးမှာ text တွေထပ်မံဖြည့်စွက်လို့ရစေပါတယ်။

PATH variable ရဲ့ content တွေအဆုံးမှာ \$HOME/bin ဆိုတဲ့ String (text လိုစာသားတွေ) ကိုပေါင်းထည့်လိုက်ခြင်းအားဖြင့် \$HOME/bin ဆိုတဲ့ directory ကို command တခုကိုရိုက်ထည့်လိုက်တဲ့အခါမှာ directory တွေကိုသွားရှာတာခံရမယ့် list ထဲမှာပေါင်းထည့်ပေးလိုက်ပါတယ်။ (shell ကလာရှာမယ့် directory list ထဲပေါင်းထည့်ပေးလိုက်တာကိုပြောတာပါ။) ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ကျွန်တော်တို့အနေနဲ့ ကျွန်တော်တို့ရဲ့ကိုယ်ပိုင် private program တွေကိုသိမ်းဆည်းဖို့အတွက် ကျွန်တော်တို့ရဲ့ home directory ထဲမှာ directory တခုကိုတည်ဆောက်တဲ့အခါမှာ shell က ကျွန်တော်တို့အတွက် ကူညီလုပ်ဆောင်ပေးဖို့အဆင်သင့်ရှိနေတယ်လို့ဆိုလိုပါတယ်။ ကျွန်တော်တို့လုပ်ရမှာကတော့ သူ့ကို bin လို့နာမည်ပေးလိုက်ပြီးတာနဲ့ကျွန်တော်တို့ကအဆင်သင့်ဖြစ်နေပြီဖြစ်ပါတယ်။

Note: distribution တော်တော်များများမှာတော့ဒီ PATH setting ကို default အနေနဲ့ပေးထားပါတယ်။ Ubuntu လိုမျိုးတချို့ Debian base distribution တွေမှာတော့ ~/bin directory ရှိမရှိကို login မှာစစ်ပေးပြီးတော့ တကယ်လို့အဲ့ဒီ directory ကိုတွေ့ခဲ့မယ်ဆိုရင် PATH variable ထဲကိုအလိုအလျောက်ပေါင်းထည့်ပေးသွားမှာဖြစ်ပါတယ်။

နောက်ဆုံးအနေနဲ့ကျွန်တော်တို့မှာဒါရှိပါသေးတယ်။

export PATH

export command က PATH ကနေဆက်သွယ်ပေးထားတဲ့ shell ရဲ့ content တွေကိုသူ့ရဲ့ child process တွေကရယူနိုင်အောင် shell ကိုပြောပြီးလုပ်ခိုင်းပါတယ်။

Modifying the Environment

ကျွန်တော်တို့အနေနဲ့ startup file တွေဘယ်နေရာမှာရှိတယ်ဆိုတာနဲ့သူတို့အထဲမှာဘာတွေပါသလဲဆိုတာကိုသိပြီးဖြစ်တာကြောင့် ကျွန်တော်တို့ရဲ့ environment ကို customize လုပ်ဖို့အတွက်သူတို့ကို modify လုပ်ရမှာဖြစ်ပါတယ်။

Which Files Should We Modify?

ပုံမှန်လုပ်နေကျအတိုင်းဆိုရင်တော့ ခင်ဗျားရဲ့ PATH ထဲမှာ directory တွေပေါင်းထည့်တာ သို့မဟုတ် နောက်ထပ် environment variable တွေထပ်ပေါင်းထည့်တာတွေလုပ်တဲ့အခါမှာအဲ့ဒီအပြောင်းအလဲလုပ်တာတွေကို .bash_profile ထဲမှာသွားပေါင်းထည့်ပေးရပါတယ်။ (ကိုယ်သုံးတဲ့ distribution အပေါ်တော့မူတည်ပါတယ်။ ဥပမာ Ubuntu ဆိုရင် .profile ဆိုတာကိုသုံးပါတယ်။) ကျန်တာတွေအားလုံးအတွက်ကတော့ အပြောင်းအလဲလုပ်တာတွေကို .bashrc ထဲမှာသွားပေါင်းထည့်ပေးရပါတယ်။ ခင်ဗျားက system administrator တယောက်ဖြစ်ပြီးတော့ system ထဲက user တွေအကုန်လုံးအတွက် default ကိုပြောင်းချင်တယ်ဆိုရင်တော့ခင်ဗျားရဲ့ home directory ထဲက file တွေကိုတော့ပြုပြင်ပြောင်းလဲခြင်းကနေ restrict ကန့်

သတ်ထားရမှာဖြစ်ပါတယ် ။ /etc ထဲက profile လို file မျိုးတွေကိုပြောင်းလဲဖို့အတွက်ဖြစ်နိုင်ပြီးတော့ကိစ္စတော်တော်များများမှာလည်းဒါကိုပြုလုပ်ဖို့အတွက်စဉ်းစားမိတတ်ကြပါတယ် ။ ဒါပေမယ့်အခုတော့ပြဿနာမဖြစ်အောင်ပဲဆော့ကြည့်ကြပါမယ် ။

Text Editors

shell ရဲ့ startup file တွေနဲ့အတူ system ပေါ်ကအခြားသော configuration file တွေကို edit လုပ်ဖို့အတွက် (တနည်းအားဖြင့် modify လုပ်ဖို့အတွက်) text editor လို့ခေါ်တဲ့ program တခုကိုအသုံးပြုပါတယ် ။ text editor ဆိုတာ program တခုဖြစ်ပြီးတော့သူက တနည်းနည်းနဲ့ word processor တခုလိုမျိုးသူက screen ပေါ်မှာ ရွေ့လျားလို့ရတဲ့ cursor တခုနဲ့အတူ စကားလုံးတွေကို edit လုပ်ခွင့်ပြုပါတယ် ။ word processor နဲ့ကွဲပြားတဲ့အချက်ကသူက pure text (စာသားသီးသန့်) ကိုသာလုပ်ပေးပြီးတော့ သူ့မှာတခါတရံ program ရေးဖို့အတွက် design လုပ်ထားတဲ့ feature တွေပါဝင်တတ်ပါတယ် ။ Text editor တွေဆိုတာ code ရေးဖို့အတွက် software developer တွေအဓိကထားသုံးတဲ့ tool ဖြစ်ပြီးတော့ system administrator တွေအနေနဲ့ကျတော့ system ကိုထိမ်းချုပ်ထားတဲ့ configuration file တွေကိုစီမံခန့်ခွဲဖို့အတွက်သုံးတာဖြစ်ပါတယ် ။

Linux အတွက် text editor ပေါင်းများစွာရှိပြီးတော့ခင်ဗျားရဲ့ system ပေါ်မှာအတော်များများ install လုပ်ပြီးသားဖြစ်ကောင်းဖြစ်နိုင်ပါတယ် ။ တချိုးထဲကိုဘာလို့အများကြီးဖြစ်နေရတာလဲ ? ဘာလို့လဲဆိုတော့ programmer တွေကအဲ့ဒါကိုရေးသားရတာနှစ်သက်ကြတဲ့အပြင် programmer တွေကအဲ့ဒါကိုကျယ်ကျယ်ပြန့်ပြန့်အသုံးပြုကြတာကြောင့် text editor တခုဆိုတာဟာဘယ်လိုမျိုးဖြစ်နေသင့်တယ်ဆိုတာကိုသူတို့ကိုယ်ပိုင်စိတ်ကြိုက်အနေအထားနဲ့ဖော်ပြချင်တာကြောင့်လည်းဖြစ်နိုင်ပါတယ် ။

text editor တွေဟာအခြေခံအားဖြင့် graphical နဲ့ text based ဆိုပြီးအမျိုးအစား ၂ မျိုးကွဲသွားပါတယ် ။ GNOME နဲ့ KDE ၂ ခုစလုံးမှာကျော်ကြားတဲ့ graphical editor တွေပါဝင်ပါတယ် ။ GNOME မှာ gedit လို့ခေါ်တဲ့ editor တပါထဲပါလာပြီးသားဖြစ်ပြီးတော့ သူ့ကိုပုံမှန်အားဖြင့် GNOME menu ထဲမှာ Text Editor လို့လည်းနာမည်ပေးထားတတ်ပါတယ် ။ KDE မှာတော့ ၃ ခုပါဝင်ပြီးတော့ သူတို့ကတော့ (အဲ့ဒါကလည်းရှုပ်ထွေးမှုကိုဖြင့်တက်စေပါတယ်) kedit , kwrite နဲ့ kate တို့ဖြစ်ကြပါတယ် ။

text-based editor တွေအများကြီးရှိပါတယ် ။ ကျော်ကြားပြီးခင်ဗျားအနေနဲ့ကြုံတွေ့ရဖို့များတာကတော့ nano , vi , နဲ့ emacs တို့ဖြစ်ကြပါတယ် ။ nano editor ကရိုးရှင်းတယ် ၊ သုံးရတာလွယ်ကူအောင်ပုံစံထုတ်ထားပြီးတော့ PINE email suite ထဲမှာပါတဲ့ pico editor ကိုအစားထိုးဖို့အတွက်ထုတ်လုပ်ထားတာဖြစ်ပါတယ် ။ vi editor (Linux system တော်တော်များများမှာသူ့ကို vim လို့အမည်ရတဲ့ program တခုနဲ့အစားထိုးကြပြီးတော့အဲ့ဒါက Vi IMproved ရဲ့အတိုကောက်လည်းဖြစ်ပါတယ် ။) ဟာ Unix-like system တွေရဲ့ ရိုးရာ editor တခုဖြစ်ပါတယ် ။ ဒါကတော့ Chapter 12 ရဲ့ဘာသာရပ်ဖြစ်ပါတယ် ။ emacs editor ကတော့ မူရင်းက Richard Stallman ရေးသားထားတာဖြစ်ပါတယ် ။ အဲ့ဒါကအလွန်ကြီးမားပြီး ၊ ဘက်စုံသုံး ၊ အကုန်လုပ်ပေးနိုင်တဲ့ programming environment တခုပဲဖြစ်ပါတယ် ။ လိုချင်ရင်အဆင်သင့်ရနိုင်ပေမယ့်လည်း Linux system တွေတော်တော်များများမှာဒါကို default အနေနဲ့ထည့်သွင်းကြတာကြုံတောင့်ကြုံခဲပါပဲ ။

Using a Text Editor

text editor တွေမှန်သမျှကို command line ကနေဖွင့်နိုင်ပြီးတော့ editor နာမည်ရဲ့အနောက်မှာ ခင်ဗျား edit လုပ်ချင်တဲ့ file ရဲ့နာမည်ကိုရိုက်ထည့်လိုက်ရုံပါပဲ ။ တကယ်လို့ file ကဆောက်ပြီးသားမရှိသေးဘူးဆိုရင် editor ကနေခင်ဗျားအနေနဲ့အသစ်ဆောက်ချင်တယ်လို့မှတ်ယူလိုက်ပါလိမ့်မယ် ။ ဒီမှာ gedit ကိုသုံးပြီးတော့ ဥပမာလုပ်ပြထားပါတယ် ။

```
jok3r@lucy:~$ gedit some_file
```

ဒီ command က gedit ကိုစတင်ပေးပြီးတော့ some_file လို့ခေါ်တဲ့ file တကယ်လို့ရှိခဲ့မယ်ဆိုရင်သူ့ကို load ခေါ်တင်ပေးပါလိမ့်မယ်။

graphical text editor မှန်သမျှဟာများသောအားဖြင့်သူ့ဟာသူရှင်းလင်းချက်တွေပါဝင်ပြီးသားဖြစ်တာကြောင့် ကျွန်တော်တို့ကဒီနေရာမှာဒါကိုမပြောတော့ပါဘူး။ ။ အဲဒီအစားကျွန်တော်တို့ကကျွန်တော်တို့ရဲ့ပထမဦးဆုံး text-based text editor ဖြစ်တဲ့ nano ကိုပဲအာရုံစိုက်ပါမယ်။ nano ကိုစတင်လိုက်ပြီးတော့ .bashrc file ကို edit လုပ်ကြစို့။ ဒါပေမယ့် ကျွန်တော်တို့အဲဒါကိုမလုပ်ခင်မှာ အရင်ဆုံး လုံခြုံစိတ်ချရတဲ့ computer ဆိုင်ရာအလေ့အကျင့်တွေကို ပြုလုပ်ပါမယ်။ ကျွန်တော်တို့အနေနဲ့ အရေးကြီးတဲ့ configuration file တွေကို edit လုပ်တဲ့အခါတိုင်းမှာ အမြဲတမ်း အရင်ဆုံး backup file တခုရဲ့ copy ကိုဖန်တီးထားတာဟာအကြံကောင်းပဲဖြစ်ပါတယ်။ ဒါကတကယ်လို့ကျွန်တော်တို့ edit လုပ်နေရင်းနဲ့ file ကို ဖျက်ဆီးမိတဲ့အခါအကူအညီပေးပါတယ်။ .bashrc file ကို backup လုပ်ဖို့အတွက်ဆိုရင် ဒီလိုလုပ်ရပါမယ်။

```
jok3r@lucy:~$ cp .bashrc .bashrc.bak
```

ခင်ဗျားအနေနဲ့ backup file ကိုဘာနာမည်ပေးတယ်ဆိုတာအရေးကြီးပါဘူး။ ခင်ဗျားနားလည်မယ့် (မှတ်မိမယ့်) နာမည်တခုကိုသာရွေးလိုက်ပါ။ .bak, .sav, .old နဲ့ .orig ဆိုတဲ့ extension တွေဟာ backup file တခုကိုညွှန်ပြရာမှာလူသိများကျော်ကြားတဲ့ extension တွေပဲဖြစ်ကြပါတယ်။ အော် ဒါနဲ့ ... cp က ရှိပြီးသား file တွေကို silently (warning မပေးဘဲ) overwrite လုပ်သွားတယ်ဆိုတာကိုမှတ်ထားပါ။

အခုကျွန်တော်တို့မှာ backup file တခုရှိသွားပြီဆိုတော့ ကျွန်တော်တို့ editor ကိုစဖွင့်ပါမယ်။

```
jok3r@lucy:~$ nano .bashrc
```

nano စတင်တာနဲ့ကျွန်တော်တို့မှာအောက်ကလို screen တခုရရှိမှာဖြစ်ပါတယ်။

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

# If not running interactively, don't do anything
case $- in
```

```
*i*) ;;  
*) return;;  
esac
```

```
# don't put duplicate lines or lines starting with space in the history.
```

```
# See bash(1) for more options
```

```
HISTCONTROL=ignoreboth
```

```
# append to the history file, don't overwrite it
```

```
shopt -s histappend
```

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

```
HISTSIZE=1000
```

```
HISTFILESIZE=2000
```

Note: တကယ်လို့ခင်ဗျားရဲ့ system ပေါ်မှာ nano မပါဘူးဆိုရင်ခင်ဗျားနေနဲ့ သူ့အစား graphical editor တခုကိုသုံးလို့ရပါတယ် ။

screen မှာ ထိပ်ဆုံးမှာ header တခုပါဝင်တယ် ၊ edit လုပ်မယ့် text တွေကအလယ်မှာ ပြီးတော့ command တွေရဲ့ menu ကအောက်ဆုံးမှာပါဝင်ပါတယ် ။ nano က email client တခုမှာပါတဲ့ text editor နေရာမှာအစားထိုးဖို့အတွက် design ထုတ်ထားတာဖြစ်တာကြောင့် သူ့မှာ editing feature တွေကနည်းပါတယ် ။

ခင်ဗျားအနေနဲ့ဘယ်လို text editor မှာမဆိုပထမဦးဆုံးသင်ကြားလေ့လာသင့်တာကတော့ program ထဲကနေ ဘယ်လိုပြန်ထွက်မလဲဆိုတာကိုပါပဲ ။ nano မှာဆိုရင်တော့ခင်ဗျားအနေနဲ့ Ctrl + X ကိုနှိပ်ပြီးထွက်လို့ရပါတယ် ။ အဲ့ဒါကို screen ရဲ့အောက်ခြေက menu မှာဖော်ပြထားပါတယ် ။ ^X လို့ notation ပြထားတဲ့သဘောက Ctrl + X ကိုရည်ညွှန်းပါတယ် ။ ဒါက control character တွေအတွက် သာမန် notation ဖြစ်ပြီးတော့ program အများအပြားကအသုံးပြုကြပါတယ် ။

ကျွန်တော်တို့အနေနဲ့သိဖို့လိုအပ်တဲ့ဒုတိယ command ကတော့ကျွန်တော်တို့လုပ်ထားတဲ့အလုပ်တွေကိုဘယ်လို save လုပ်မလဲဆိုတာပဲဖြစ်ပါတယ် ။ nano နဲ့ဆိုရင်တော့ အဲ့ဒါက CTRL -O (အို) ဖြစ်ပါတယ် ။ ဒီပညာတွေကိုကျွန်တော်တို့ ကပိုင်ပိုင်နိုင်နိုင်သိသွားပြီဆိုရင်တော့ ကျွန်တော်တို့က editing လုပ်ဖို့အဆင်သင့်ဖြစ်ပါပြီ ။ down-arrow key နဲ့/သို့မဟုတ် page-down key ကိုအသုံးပြုပြီးတော့ cursor ကို file ရဲ့အဆုံးကိုရွှေ့ပြီးတော့ .bashrc file ထဲမှာအောက်ပါ line (စာကြောင်း) တွေကိုရေးထည့်လိုက်ပါ ။

```
umask 0002
```

```
export HISTCONTROL=ignoredups
```

```
export HISTSIZE=1000
```

```
alias l.='ls -d .* --color=auto'
```

```
alias ll='ls -l --color=auto'
```

Note: ခင်ဗျားရဲ့ distribution ထဲမှာဒီထဲကဟာတချို့ပါပြီးသားဖြစ်ကောင်းဖြစ်နိုင်ပေမယ့် ၂ ခါထပ်သွားလည်းဘာမှမထိခိုက်ပါဘူး ။

Table 11-4 မှာကျွန်တော်တို့ထပ်ရေးထည့်လိုက်တဲ့စာတွေရဲ့အဓိပ္ပါယ်တွေကို list နဲ့ပြထားပါတယ် ။

Table 11-4: Additions to Our .bashrc File

Line	Meaning
Umask 0002	umask ကို set လုပ်ပြီးတော့ကျွန်တော်တို့ chapter 90 မှာတုန်းကဆွေးနွေးခဲ့ကြတဲ့ share directory ပြဿနာကိုဖြေရှင်းလိုက်တာပါ ။
export HISTCONTROL=ignoredups	command တခုကိုအခုပဲ record လုပ်ပြီးသားရှိနေခဲ့မယ်ဆိုရင် သူနဲ့တူညီတဲ့နောက်ထပ် command တခုကို(ထပ်ရိုက်ထည့်တဲ့အခါ record မလုပ်ပဲ) ignore လုပ်ဖို့ shell ရဲ့ history recording feature ကိုခိုင်းလိုက်တာပါ ။
export HISTSIZE=1000	command history ရဲ့ size ကို default 500 lines ကနေ 1000 lines ဖြစ်အောင်တိုးခိုင်းလိုက်တာပါ ။
alias l='ls -d .* --color=auto'	dot (.) နဲ့စတဲ့ directory entry တွေအကုန်လုံးကိုဖော်ပြပေးတဲ့ l. (အယ်လ် နဲ့အစက်တစက်) ဆိုတဲ့ command အသစ်တခုဖန်တီးလိုက်တာပါ ။
alias ll='ls -l --color=auto'	long-format directory listing တခုကိုဖော်ပြပေးတဲ့ ll (အယ်လ် ၂ လုံး) ဆိုတဲ့ command အသစ်တခုဖန်တီးလိုက်တာပါ ။

ကျွန်တော်တို့မြင်နေရတဲ့အတိုင်းပဲ များစွာသောကျွန်တော်တို့ရဲ့ထပ်ပေါင်းထည့်မှုတွေဟာ အလိုလိုသိသာထင်ရှားမနေပါဘူး ။ ဒါကြောင့်ကျွန်တော်တို့ရဲ့ .bashrc file ထဲမှာ လူသားတွေကိုဒီဟာတွေကိုရှင်းပြဖို့အတွက် comment အနည်းငယ်ထည့်ပေးခြင်းကကောင်းတဲ့အကြံပဲဖြစ်ပါတယ် ။ editor ကိုအသုံးပြုပြီးတော့ကျွန်တော်တို့ထပ်ထည့်ထားတာတွေကိုဒီလိုပုံစံမျိုးဖြစ်အောင်လုပ်ပါမယ် ။

```
# Change umask to make directory sharing easier
umask 0002

# Ignore duplicates in command history and increase
# history size to 1000 lines
export HISTCONTROL=ignoredups
export HISTSIZE=1000

# Add some helpful aliases
alias l='ls -d .* --color=auto'
alias ll='ls -l --color=auto'
```

အင်း .. ပိုကောင်းသွားပြီ ။ ကျွန်တော်တို့အပြောင်းအလဲလုပ်တာပြီးပြီဆိုရင် CTRL -O ကိုနှိပ်ပြီးတော့ကျွန်တော်တို့ modify လုပ်ထားတဲ့ .bashrc file လေးကို save ပြီး CTRL -X နဲ့ nano ထဲကနေထွက်ပါမယ် ။

Activating Our Changes

ကျွန်တော်တို့ရဲ့ .bashrc မှာပြုလုပ်ထားတဲ့အပြောင်းအလဲတွေဟာ ကျွန်တော်တို့က ကျွန်တော်တို့ရဲ့ terminal session ကိုပိတ်ပြီးအသစ်တစ်ခုပြန်မစမချင်း effect မဖြစ်ပါဘူး ။ ဘာလို့လဲဆိုတော့ .bashrc file ကို session တစ်ခုရဲ့အစပိုင်းမှာမှဖတ်ပေးလို့ဖြစ်ပါတယ် ။ ဘယ်လိုပဲဖြစ်ဖြစ်ကျွန်တော်တို့အနေနဲ့ bash ကို force လုပ်ပြီးတော့ .bashrc file ကိုအောက်ပါ command နဲ့ပြန်ဖတ်ခိုင်းလို့ရပါတယ် ။

```
jok3r@lucy:~$ source .bashrc
```

အဲဒါကိုလုပ်ပြီးတဲ့နောက်မှာ ကျွန်တော်တို့အနေနဲ့ကျွန်တော်တို့ပြုလုပ်ခဲ့တဲ့အပြောင်းအလဲကိုမြင်တွေ့ရသင့်ပါတယ် ။ ကျွန်တော်တို့အသစ်ပြုလုပ်ခဲ့တဲ့ alias တွေထဲကတစ်ခုကိုစမ်းကြည့်လိုက်ပါ ။

```
jok3r@lucy:~$ ll
```

WHY COMMENTS ARE IMPORTANT

ခင်ဗျားအနေနဲ့ configuration file တွေကို modify လုပ်တဲ့အချိန်တိုင်းမှာ ခင်ဗျားအပြောင်းအလဲလုပ်တဲ့ document မှာ comment တွေထည့်ပေးခြင်းဟာအကြံကောင်းဖြစ်ပါတယ် ။ ဟုတ်ပါတယ် ၊ ခင်ဗျားအခုလုပ်ထားတဲ့ အပြောင်းအလဲတွေကိုမနက်ဖန်ကျရင်မှတ်မိနေမှာသေချာတာပေါ့ ။ ဒါပေမယ့်အခုကနေစပြီးနောက် ၆ လလောက် ကြာတဲ့အခါကျရင်ရော ? ခင်ဗျားကိုခင်ဗျားအခွင့်အရေးပေးတဲ့အနေနဲ့ comment အနည်းငယ်ရေးထားလိုက်ပါ ။ ဒါနဲ့ တဆက်ထဲ ခင်ဗျားလုပ်ခဲ့တဲ့အပြောင်းအလဲတွေကို log တစ်ခုမှတ်ထားတာကဆိုးတဲ့အကြံတော့မဟုတ်ပါဘူး ။

Shell scripts နဲ့ bash startup file တွေမှာ comment စရေးဖို့အတွက် # ကိုအသုံးပြုပါတယ် ။ တခြား configuration file တွေကတော့တခြား symbol တွေကိုအသုံးပြုကြပါတယ် ။ configuration file တွေအများစုမှာ comment တွေပါပြီးတော့အဲဒါတွေကို guide အနေနဲ့အသုံးပြုပါ ။

ခင်ဗျားအနေနဲ့တခါတလေမှာ configuration file ထဲက line တွေကိုလက်ရှိ program က အသုံးပြုခြင်း (ဖတ်ပြီး run ခြင်း) မှကာကွယ်ဖို့အတွက် commented out (စာကြောင်းအစမှာ # တွေထည့်ပြီး code တွေကို comment အဖြစ်ပုံဖျက်ထားတာ) လုပ်ထားတာကိုတွေ့ရမှာပါ ။ ဒါကိုလုပ်ရတဲ့အကြောင်းက ဒါကိုဖတ်တဲ့လူကို ဖြစ်နိုင်ခြေရှိတဲ့ configuration choice တွေအကြံပြုတာ သို့မဟုတ် မှန်ကန်တဲ့ configuration syntax တွေကိုဥပမာ အနေနဲ့ပြပေးတာမျိုးဖြစ်ပါတယ် ။ ဥပမာ - Ubuntu 20.04 ရဲ့ .bashrc file မှာဆိုရင်ဒီလိုမျိုး line တွေပါဝင်နေပါတယ် ။

```
# some more ls aliases
#alias ll='ls -aF'
#alias la='ls -A'
#alias ll='ls -aF'
#alias la='ls -A'
#alias l='ls -CF'
```

နောက်ဆုံး ၃ ကြောင်းက valid alias definition တွေဖြစ်ပြီးတော့ သူတို့ကို commented out လုပ်ထားပါတယ်။ ခင်ဗျားအနေနဲ့အဲဒီစာ ၃ ကြောင်းအရှေ့က # symbol တွေကိုဖယ်လိုက်မယ်ဆိုရင် နည်းပညာအရဒါကို uncommenting လို့ခေါ်ပြီးတော့ခင်ဗျားအနေနဲ့ alias တွေကို activate လုပ်လိုက်တာပါပဲ။ ပြောင်းပြန်အနေနဲ့ ခင်ဗျားကစာကြောင်းတခုရဲ့အစမှာ # symbol တခုကိုထည့်လိုက်မယ်ဆိုရင် ခင်ဗျားအနေနဲ့သူ့ထဲမှာပါတဲ့ information ကိုထိမ်းသိမ်းလိုက်ပြီး configuration line ကို deactivate လုပ်လိုက်တာဖြစ်ပါတယ်။

Final Note

အခု Chapter မှာ ကျွန်တော်တို့အနေနဲ့မရှိမဖြစ်လိုအပ်တဲ့ skill တခုဖြစ်တဲ့ configuration file တွေကို text editor နဲ့ editing လုပ်တာကိုလေ့လာသင်ကြားခဲ့ပြီးဖြစ်ပါတယ်။ ရှေ့ဆက်ပြီးတော့ကျွန်တော်တို့က command ရဲ့ man page တွေကိုဖတ်သွားရင်းနဲ့ command တွေက support လုပ်တဲ့ environment variable တွေကိုမှတ်သားထားပါမယ်။ အဲဒီမှာ အဖိုးတန်တာတခုနှစ်ခုရှိနိုင်ပါတယ်။ နောက်ပိုင်း chapter တွေမှာကျွန်တော်တို့က shell function တွေအကြောင်းကိုလေ့လာသွားမှာဖြစ်ပြီးတော့ (အဲဒါက) bash startup file ထဲမှာထည့်ပေါင်းလို့ရတဲ့ အရမ်းအစွမ်းထက်တဲ့ feature တခုကိုခင်ဗျားရဲ့ custom command လက်နက်တိုက်ထဲမှာထည့်ပေါင်းလို့ရသွားမှာဖြစ်ပါတယ်။

A GENTLE INTRODUCTION TO VI

New York မြို့ကိုလာလည်တဲ့ဧည့်သည်တယောက်က ဘေးကဖြတ်သွားတဲ့လူတယောက်ကို မြို့ရဲ့နာမည်ကြီး classical music ကျင်းပရာနေရာကိုမေးတာနဲ့ပတ်သက်ပြီးတော့ ဟာသအဟောင်းတခုရှိပါတယ်။ ။

ဧည့်သည် - တဆိတ်လောက်ခင်ဗျာ ။ ကျွန်တော် Carnegie Hall ထဲရောက်အောင်ဘယ်လိုသွားရမလဲဗျာ ?
ဖြတ်သွားသူ - လေ့ကျင့် ၊ လေ့ကျင့် ၊ လေ့ကျင့် !

Linux command line ကိုလေ့လာခြင်းဟာ စန္ဒယားဆရာတယောက်ဖြစ်အောင်လုပ်တာနဲ့အတူတူပါပဲ ။ ဒါက တညနေထဲလောက်နဲ့ကောက်ကာငင်ကာဖြစ်လာတဲ့အရာတွေမဟုတ်ပါဘူး ။ နှစ်ပေါင်းများစွာလေ့ကျင့်မှုလိုအပ်ပါတယ် ။ အခု chapter မှာကျွန်တော်တို့က Unix ရိုးရာထဲက program တခုဖြစ်တဲ့ vi (“vee eye” ဗီးအိုင်း လို့အသံထွက်တယ်) ဆိုတဲ့ text editor နဲ့မိတ်ဆက်ပေးမှာဖြစ်ပါတယ် ။ vi ဟာသုံးရခက်တဲ့ user interface ကြောင့်တော်တော်လေးနာမည်ဆိုးနဲ့ကျော်ကြားပေမယ့်လည်းကျွန်တော်တို့အနေနဲ့ဆရာသမားက (စန္ဒယား)keyboard မှာထိုင်ပြီးတော့ပျော်ဖြေတဲ့အခါကျတော့လည်းကျွန်တော်တို့အနေနဲ့တကယ့်ခမ်းနားတဲ့အနုပညာကိုမျက်မြင်တွေ့ခွင့်ရပါလိမ့်မယ် ။ ကျွန်တော်တို့အနေနဲ့ယခု chapter မှာ ဆရာကြီးဖြစ်သွားမှာမဟုတ်ပါဘူး ။ ဒါပေမယ့်ကျွန်တော်တို့ပြီးသွားတဲ့အခါကျရင်တော့ ကျွန်တော်တို့အနေနဲ့ vi ထဲမှာ “Chopsticks” (classical waltz သီချင်း) ကိုဘယ်လိုတီးရတယ်ဆိုတာကိုသိသွားမှာဖြစ်ပါတယ် ။ (ကျွမ်းကျင်သွားမယ်လို့ဆိုလိုတာပါ)

Why We Should Learn vi

graphical editor တွေနဲ့ nano လိုမျိုးသုံးရလွယ်ကူတဲ့ text-based editor တွေရှိနေတဲ့အခုလိုခေတ်မှီနေတဲ့ခေတ်ကြီးထဲမှာ ကျွန်တော်တို့အနေနဲ့ vi ကိုလာလို့လေ့လာဖို့လိုတာလဲ ? အကြောင်းပြချင်ကောင်းကောင်း ၃ ခုရှိပါတယ် ။

- vi ကအမြဲတမ်းရနိုင်ပါတယ် ။ ဒါကအသက်ကယ်ဆေးတခုဖြစ်လာနိုင်ပါတယ် တကယ်လို့ကျွန်တော်တို့မှာ remote server တခုလို သို့မဟုတ် X configuration ပျက်နေတဲ့ local system တခုလိုမျိုး graphical interface မပါတဲ့ system တခုရှိနေမယ်ဆိုရင်ပေါ့ ။ nano ကကျော်ကြားမှုတိုးလာပေမယ့်လည်း သူက Universal မဖြစ်သေးပါဘူး ။ Unix system တွေပေါ်မှာ program compatibility ဖြစ်ဖို့အတွက် standard တခုဖြစ်တဲ့ POSIX အနေနဲ့ vi ရှိနေဖို့လိုအပ်ပါတယ် ။
- vi ဟာပေါ့ပါးပြီးမြန်ဆန်ပါတယ် ။ ကိစ္စတော်တော်များများမှာ menu တွေထဲကနေ graphical text editor ကိုရှာပြီး သူ့ရဲ့ Megabyte အများကြီးတက်တာကိုစောင့်နေရတာထက်စာရင် vi ကိုခေါ်လိုက်တာကပိုလွယ်ကူပါတယ် ။ နော်ကတခုက vi က typing speed အတွက် design ထုတ်ထားတာဖြစ်ပါတယ် ။ ကျွန်တော်တို့နောက်ဆိုလည်းမြင်ရမှာဖြစ်တဲ့အတိုင်း ကျွမ်းကျင်တဲ့ vi user တယောက်အနေနဲ့ editing လုပ်နေချိန်မှာ သူ သို့မဟုတ် သူမ ရဲ့ လက်ချောင်းလေးတွေကို keyboard ကနေခွာစရာကိုမလိုပါဘူး ။ (mouse မသုံးပဲလုပ်လို့ရတယ်လို့ဆိုလိုပါတယ်)
- တခြား Linux သို့မဟုတ် Unix user တွေကကျွန်တော်တို့ကို အခြောက်တွေ(cကြောက်တွေ) လို့မထင်စေချင်ပါဘူး ။ ကောင်းပါပြီ ။ အကြောင်းပြချက်ကောင်းကောင်း ၂ ခုလို့ပဲထားလိုက်ပါတော့ ။ :P

A Little Background

ပထမဦးဆုံးသော vi version ကို ၁၉၇၆ မှာ Bill Joy ကရေးသားခဲ့ပါတယ်။ California တက္ကသိုလ်တွေထဲကတစ်ခုဖြစ်တဲ့ Berkeley ကျောင်းသားဖြစ်ပြီးတော့ နောက်ပိုင်းမှာ Sun Microsystems ကိုပူးတွဲတည်ထောင်သူ ဖြစ်လာပါတယ်။ vi ဆိုတဲ့နာမည်က visual ဆိုတဲ့စကားလုံးကနေဆင်းသက်လာတာပါ။ ဘာကြောင့်လဲဆိုတော့သူက လှုပ်ရှားလို့ရတဲ့ cursor တခုနဲ့ video terminal တခုပေါ်မှာ editing လုပ်ဖို့အတွက်ရည်ရွယ်ထားတာကြောင့်ဖြစ်ပါတယ်။ visual editor တွေမတိုင်ခင်က line editor တွေပဲရှိပြီးတော့သူတို့က တကြိမ်မှာ စာတကြောင်းပေါ်မှာပဲ operate လုပ်ဆောင်ပေးနိုင်ကြပါတယ်။ အပြောင်းအလဲတခုလုပ်ချင်တဲ့အခါမှာကျွန်တော်တို့က line editor ကို ပြောင်းလဲချင်တဲ့ line တခုဆီကိုသွားခိုင်းပြီးတော့ ဘာတွေပြောင်းလဲပေးရမလဲဆိုတာကိုပြောပြပေးရပါတယ်။ စာတွေထပ်ထည့်တာ သို့မဟုတ် စာတွေဖျက်ပစ်တာလိုမျိုးတွေကိုပေါ့။ video terminal ရဲ့အားသာချက်ကြောင့် (teletype တွေလိုမျိုး printer-based terminal တွေထက်စာရင်သူကအားသာပါတယ်။ အဲ့အချိန်က) visual editing လုပ်လို့ရသွားပါတယ်။ vi က တကယ်တော့ ex လို့ခေါ်တဲ့ powerful line editor နဲ့တရားဝင်ပေါင်းစပ်ထားတာကြောင့် ကျွန်တော်တို့အနေနဲ့ vi ကိုသုံးနေစဉ်မှာ line-editing command တွေကိုအသုံးပြုလို့ရသွားပါတယ်။

Starting and Stopping vi

vi ကိုစဖွင့်ဖို့အတွက်ကျွန်တော်တို့အနေနဲ့ ရိုးရိုးလေးအောက်ပါအတိုင်းရိုက်ထည့်လိုက်ရုံပါပဲ။

```
jok3r@lucy:~$ vi
```

```
ဒီလိုပုံစံမျိုး screen တခုပေါ်လာပါလိမ့်မယ်။
```

```
~
~
~      VIM - Vi IMproved
~
~      version 8.1.3741
~      by Bram Moolenaar et al.
~      Modified by team+vim@tracker.debian.org
~      Vim is open source and freely distributable
~
~      Sponsor Vim development!
~      type :help sponsor<Enter>   for information
~
~      type :q<Enter>               to exit
~      type :help<Enter> or <F1>    for on-line help
~      type :help version8<Enter>  for version info
```

~
~ Running in Vi compatible mode
~ type :set nocp<Enter> for Vim defaults
~ type :help cp-default<Enter> for info on this
~
~
~

ကျွန်တော်တို့အစောပိုင်းတုန်းက nano မှာလုပ်ခဲ့သလိုပဲ ပထမဦးဆုံးလေ့လာရမှာကတော့ဘယ်လိုပြန်ထွက်ရမလဲဆိုတာပဲဖြစ်ပါတယ်။ ပြန်ထွက်ဖို့အတွက် ကျွန်တော်တို့ကအောက်ပါ command ကိုရိုက်ထည့်ပေးရမှာဖြစ်ပါတယ်။ (တခုမှတ်ထားရမှာက colon character (:)) ဟာ command ရဲ့အစိတ်အပိုင်းတခုဖြစ်ပါတယ်။

:q

shell prompt q ဆီကိုပြန်ရောက်သွားပါလိမ့်မယ်။ တကယ်လို့ ၊ အခြားအကြောင်းတွေကြောင့် vi ကပြန်မထွက်သွားခဲ့ရင် (ပုံမှန်အားဖြင့် file တခုမှာအပြောင်းအလဲတွေလုပ်ပြီးတော့ မ save ခဲ့လို့) ကျွန်တော်တို့အနေနဲ့ vi ကိုကျွန်တော်တို့ (ထွက်ဖို့) တကယ်ပြောနေတာပါဆိုတဲ့အကြောင်းကို command မှာ exclamation point (!) တခုထည့်ပြီးပြောလို့ရပါတယ်။

:q!

Note: တကယ်လို့ခင်ဗျားအနေနဲ့ vi ထဲမှာလမ်းပျောက်နေခဲ့ရင် ESC key ကို ၂ ချက်နှိပ်ပြီးတော့ခင်ဗျားရဲ့လမ်းကိုရှာဖွေနိုင်ပါတယ်။

Editing Modes

vi ကိုနောက်တခါပြန်ဖွင့်ကြစို့။ ဒီတခါမှာတော့သူ့ဆီကို မရှိတဲ့ (မဆောက်ရသေးတဲ့) file တခုရဲ့နာမည်ကို passing ပေးကြည့်လိုက်ပါမယ်။ ဒီလိုနည်းနဲ့ကျွန်တော်တို့က file အသစ်တခုကို vi နဲ့တည်ဆောက်လို့ရပါတယ်။

jok3r@lucy:~\$ rm -f foo.txt

jok3r@lucy:~\$ vi foo.txt

တကယ်လို့အကုန်လုံးအဆင်ပြေပြေဖြစ်သွားမယ်ဆိုရင်ကျွန်တော်တို့အနေနဲ့ဒီလို screen မျိုးကိုမြင်တွေ့ရမှာဖြစ်ပါတယ်။

```
"foo.txt" [New File]
```

vi နဲ့ပတ်သက်ပြီးလေ့လာဖို့ဒုတိယအရေးအကြီးဆုံးအရာကတော့ (vi ထဲကပြန်ထွက်တာကိုလေ့လာပြီးတဲ့နောက်မှာ) vi ဟာ modal editor တခုဖြစ်တယ်ဆိုတာကိုပါ။ vi ကိုစဖွင့်လိုက်တဲ့အခါမှာသူက command mode ထဲမှာစတင်ပေးပါတယ်။ အဲ့ဒီ mode ထဲမှာ key တိုင်းဟာ command တခုစီဖြစ်တာကြောင့် ကျွန်တော်တို့ကစာစရိုက်လိုက်မယ်ဆိုရင်အခြေခံအားဖြင့် vi ဟာ ရူးသွားပြီးတော့ ရှုပ်ထွေးမှုကြီးကြီးမားမားတွေကို လုပ်လိုက်မှာဖြစ်ပါတယ်။

ကျွန်တော်တို့ရဲ့ file ထဲကို စာသားတချို့ထည့်ဖို့အတွက်ဆိုရင် ကျွန်တော်တို့အနေနဲ့ပထမဦးဆုံး insert mode ထဲကို ဝင်ရပါမယ် ။ ဒါကိုလုပ်ဖို့အတွက်ကျွန်တော်တို့က I key (i) ကိုနှိပ်ရပါမယ် ။ ပြီးရင်တော့ကျွန်တော်တို့အနေနဲ့ screen ရဲ့ အောက်ခြေနားမှာအောက်ပါအတိုင်းမြင်တွေ့ရမှာဖြစ်ပါတယ် ။ တကယ်လို့ vim ကသူ့ရဲ့ ပုံမှန် enhanced mode မှာ run နေခဲ့မယ်ဆိုရင်ပေါ့ (vi -compatible mode မှာတော့ဒါကပေါ်လာမှာမဟုတ်ပါဘူး ။)

အခုဆိုရင်ကျွန်တော်တို့ စာရိုက်လို့ရပါပြီ ။ ဒါလေးကိုရိုက်ကြည့်လိုက်ပါ ။

The quick brown fox jumped over the lazy dog.

insert mode ကနေထွက်ပြီးတော့ command mode ကိုပြန်သွားဖို့အတွက် Esc Key ကိုနှိပ်ပါ။

Saving Our Work

ကျွန်တော်တို့ရဲ့ file မှာပြုလုပ်ခဲ့တဲ့အပြောင်းအလဲတွေကို save လုပ်ဖို့အတွက် ကျွန်တော်တို့အနေနဲ့ command mode ထဲမှာရှိနေစဉ်မှာ ex command ကိုရိုက်ထည့်ပေးရပါမယ်။ ဒါကိုအလွယ်တကူပဲ : key ရိုက်ထည့်လိုက်ပြီးတော့လုပ်လို့ရပါတယ်။ ဒါကိုလုပ်ပြီးတဲ့နောက်မှာ screen ရဲ့အောက်ခြေနားမှာ colon character (:) ပေါ်နေရမှာဖြစ်ပါတယ်။

:

ကျွန်တော်တို့ အပြောင်းအလဲလုပ်ထားတဲ့ file ထဲမှာသွားရေးစေဖို့အတွက် : ရဲ့အနောက်မှာ w တလုံးရိုက်ထည့်ပြီးတော့ Enter နှိပ်လိုက်ပါ။

:w

file ကို hard drive ထဲမှာသွားရေးလိုက်မှာဖြစ်ပြီးတော့ screen ရဲ့အောက်ခြေနားမှာကျွန်တော်တို့အတွက် ဒီလို confirmation message တခုရရှိမှာဖြစ်ပါတယ်။

"foo.txt" [New File] 1 line, 43 characters written

Note: တကယ်လို့ခင်ဗျားအနေနဲ့ vim documentation ကိုဖတ်ကြည့်မယ်ဆိုရင်ခင်ဗျားအနေနဲ့ (ရှုပ်ထွေးလှစွာနဲ့) သတိထားမိမှာကတော့ command mode ကို normal mode လို့ခေါ်ပြီးတော့ ex command တွေကိုကျတော့ command mode လို့ခေါ်တယ်ဆိုတာကိုပါပဲ။

COMPATIBILITY MODE

ဥပမာထဲမှာ ယခု section ရဲ့အစမှာပြထားတဲ့ startup screen (Ubuntu 20.04 က ယူထားတာ) မှာ text တွေက Vi compatible mode ထဲမှာ run နေတာကိုတွေ့ရပါမယ်။ ဒါကဘာကိုဆိုလိုတာလဲဆိုတော့ vim ဟာ သူ့ရဲ့ enhanced behavior ထက်စာရင် vi ရဲ့ normal behavior နဲ့နီးစပ်တဲ့ mode ထဲမှာ run ပေးမယ်လို့ဆိုလိုတာဖြစ်ပါတယ်။ အခုဒီ chapter အတွက်တော့ကျွန်တော်တို့အနေနဲ့ vim ကိုသူ့ရဲ့ enhanced behavior နဲ့ run ပါမယ်။ ဒါကိုလုပ်ဖို့အတွက်ခင်ဗျားအနေနဲ့ရွေးချယ်စရာတွေရှိပါတယ်။

- vi အစား vim ကို run ကြည့်ပါ (တကယ်လို့အလုပ်ဖြစ်ခဲ့မယ်ဆိုရင် ခင်ဗျားရဲ့ .bashrc file ထဲမှာ vi='vim' ဆိုပြီး alias ပေါင်းထည့်လိုက်ပါ)
- ဒီ command ကိုအသုံးပြုပြီး vim configuration file ထဲကိုစာတကြောင်းပေါင်းထည့်လိုက်ပါ။

echo "set nocp" >> ~/.vimrc

မတူညီတဲ့ Linux distribution package မှာ vim လုပ်ဖို့အတွက်မတူညီတဲ့နည်းလမ်းတွေရှိပါတယ်။ တချို့ distribution တွေမှာ vim ရဲ့ minimal version တခုကိုသာ default အနေနဲ့ထည့်သွင်းထားပေးပြီးအဲ့ဒါက လည်း ကန့်သတ်ထားတဲ့ vim feature set တခုကိုပဲ support ပေးပါတယ်။ နောက်ပိုင်းမှာပါလာမယ့်သင်ခန်းစာ တွေကိုလုပ်ရင်းနဲ့ခင်ဗျားအနေနဲ့ပျောက်နေတဲ့ (ပါမလာတဲ့) feature တွေကိုကြုံတွေ့ရနိုင်ပါတယ်။ အဲ့ဒီလိုဖြစ်ခဲ့ရင် vim ရဲ့ full version ကိုသာ install လုပ်လိုက်ပါ။

Moving the Cursor Around

command mode ထဲမှာရှိနေစဉ်မှာ vi က ရွှေ့လျားမှုနဲ့ပတ်သက်တဲ့ command တွေအများကြီးပေးထား ပြီးတော့တချို့ဟာတွေက less နဲ့မျှသုံးနေတာဖြစ်ပါတယ်။ Table 12-1 မှာ subset တွေကို list ထုတ်ပြထားပါတယ်။

Table 12-1: Cursor Movement Keys

Key	Moves the cursor
L or right arrow	character တလုံးစာညာဖက်ကိုရွှေ့ပေးမယ်။
H or left arrow	character တလုံးစာဗယ်ဖက်ကိုရွှေ့ပေးမယ်။
J or down arrow	လိုင်းတလိုင်းဆင်းပေးမယ်။
K or up arrow	လိုင်းတလိုင်းတက်ပေးမယ်။
0 (zero)	လက်ရှိလိုင်းရဲ့အစ(ဗယ်ဖက်ထိပ်ဆုံး)ကိုသွားပေးမယ်။
SHIFT -6 (^)	လက်ရှိလိုင်းရဲ့ ပထမဦးဆုံး whitespace character ဆီသွားပေးမယ်။
SHIFT -4 (\$)	လက်ရှိလိုင်းရဲ့အဆုံး(ညာဖက်အဆုံး)ကိုသွားပေးမယ်။
W	နောက်ထပ်စကားလုံး သို့မဟုတ် punctuation character ရဲ့အစကိုသွားပေးမယ်။
SHIFT -W (W)	punctuation character ကိုထည့်မတွက်ဘဲ နောက်ထပ်စကားလုံးရဲ့အစကိုသွားပေးမယ်။
B	ပြီးခဲ့တဲ့ စကားလုံး သို့မဟုတ် punctuation character ရဲ့အစကိုသွားပေးမယ်။
SHIFT -B (B)	punctuation character ကိုထည့်မတွက်ဘဲ ပြီးခဲ့တဲ့ စကားလုံးရဲ့အစကိုသွားပေးမယ်။
CTRL -F or PAGE DOWN	စာတမျက်နှာစာအောက်ဆင်းပေးမယ်။
CTRL -B or PAGE UP	စာတမျက်နှာစာအပေါ်တက်ပေးမယ်။
number- SHIFT -G	ပေးထားတဲ့လိုင်းနံပါတ်ဆီကိုသွားပေးမယ်။ (ဥပမာ - 1G ဆိုရင် file ရဲ့ပထမဦးဆုံးလိုင်း

	ဆီကိုသွားပေးမယ်။)
SHIFT -G (G)	file ရဲ့နောက်ဆုံးလိုင်းဆီကိုသွားပေးမယ်။

cursor movement အတွက်ဘာကြောင့် H, J, K နဲ့ L key တွေကိုအသုံးပြုနေရတာလဲ ? ဘာလို့လဲဆိုတော့ vi ကိုစပြီးရေးစဉ်အခါတုန်းက video terminal တွေအကုန်လုံးမှာ Arrow key တွေမပါဝင်သေးပါဘူး။ ပြီးတော့စာရိုက်ကျွမ်းကျင်တဲ့သူတွေဟာ သူတို့ရဲ့လက်ချောင်လေးတွေကို keyboard ကနေရွှေ့စရာမလိုပဲပုံမှန် keyboard က key တွေကိုသုံးပြီးတော့ cursor ရွှေ့ဖို့အတွက်အသုံးပြုနိုင်လို့ပဲဖြစ်ပါတယ်။

vi ထဲက command အများအပြားကို နံပါတ်တစ်ခုနဲ့ prefix လုပ်လို့ရပါတယ်။ Table 12-1 မှာ G command အတွက်ပြထားသလိုပါပဲ။ command တစ်ခုကို နံပါတ်တစ်ခုနဲ့ prefix လုပ်လိုက်ခြင်းအားဖြင့်ကျွန်တော်တို့အနေနဲ့ command တစ်ခုကိုဘယ်နှစ်ကြိမ်ဆက်လုပ်ပါလို့ အတိအကျပြောလိုက်သလိုပါပဲ။ ဥပမာ 5j ဆိုတဲ့ command ဆိုရင် vi ကို အောက် ၅ လိုင်းအထိ cursor ဆင်းသွားခိုင်းတာပါပဲ။

Basic Editing

editing တော်တော်များများမှာ စာထပ်ထည့်တာ၊ စာဖျက်တာ နဲ့ text တွေကို cut နဲ့ paste လုပ်ပြီးဟိုရွှေ့ဒီရွှေ့လုပ်တာလိုမျိုးအခြေခံ basic operation တချို့ပါဝင်ပါတယ်။ vi ကလည်း ဟုတ်ပါတယ်။ ဒီလုပ်ဆောင်ချက်တွေကိုသူ့ရဲ့ကိုယ်ပိုင်ရှုးပါးတဲ့ပုံစံနဲ့လုပ်ဆောင်ပေးပါတယ်။ vi က undo ကိုလည်းပဲကန့်သတ်ထားတဲ့ပုံစံတစ်ခုနဲ့လုပ်ပေးပါတယ်။ တကယ်လို့ကျွန်တော်တို့က command mode ထဲမှာရှိနေစဉ်မှာ U key ကိုနှိပ်လိုက်မယ်ဆိုရင် vi ကခင်ဗျားနောက်ဆုံးပြုလုပ်ခဲ့တဲ့ပြောင်းလဲမှုကို undo လုပ်ပေးမှာဖြစ်ပါတယ်။ ဒါကကျွန်တော်တို့ အခြေခံ editing command တွေကိုစမ်းလုပ်ကြည့်တဲ့အခါမှာအသုံးဝင်လာမှာဖြစ်ပါတယ်။

Appending Text

vi မှာ insert mode ထဲဝင်ဖို့အတွက်နည်းလမ်းများစွာရှိပါတယ်။ စာတွေထည့်ဖို့အတွက်ကျွန်တော်တို့ i command ကိုအသုံးပြုခဲ့ပြီးဖြစ်ပါတယ်။ ကျွန်တော်တို့ရဲ့ foo.txt file ဆီကိုခဏလောက်ပြန်သွားကြစို့။

The quick brown fox jumped over the lazy dog.

တကယ်လို့ ဒီစာကြောင်းရဲ့အဆုံးမှာကျွန်တော်တို့က စာတချို့ထပ်ပေါင်းထည့်ချင်တယ်ဆိုရင်ကျွန်တော်တို့အနေနဲ့ i command ကလုပ်ပေးမှာမဟုတ်ဘူးဆိုတာကိုရှာဖွေတွေ့ရှိလာမှာပါ။ ဘာဖြစ်လို့လဲဆိုတော့ကျွန်တော်တို့က cursor ကိုလိုင်းရဲ့အဆုံးထက်ကျော်လွန်ပြီးသွားအောင်မလုပ်နိုင်လို့ဖြစ်ပါတယ်။ vi ကစာတွေထပ်ထည့်ဖို့အတွက် command တစ်ခုကိုထည့်သွင်း ပေးထားပါတယ်။ ထင်သာမြင်သာတဲ့နာမည်ဖြစ်တဲ့ a command ဖြစ်ပါတယ်။ ကျွန်တော်တို့အနေနဲ့ cursor ကိုစာကြောင်းအဆုံးနားမှာထားပြီးတော့ a ကိုရိုက်ထည့်လိုက်မယ်ဆိုရင် cursor ကစာကြောင်းအဆုံးကိုကျော်လွန်သွားပြီးတော့ vi က insert mode ထဲကိုဝင်သွားမှာဖြစ်ပါတယ်။ ဒါကကျွန်တော်တို့ကိုစာတွေထပ်ထည့်လို့ရသွားစေပါတယ်။

The quick brown fox jumped over the lazy dog. **It was cool** .

insert mode ထဲကထွက်ဖို့အတွက် Esc key ကိုနှိပ်ဖို့မမေ့ပါနဲ့။

ကျွန်တော်တို့တွေက စာကြောင်းရဲ့အဆုံးမှာစာတွေကိုအမြဲတမ်းလိုလိုထပ်ထပ်ထည့်နေမှာဖြစ်တာကြောင့် vi က လက်ရှိစာကြောင်းရဲ့အဆုံးကို cursor ကိုရွှေ့သွားပြီးတော့ စာတွေထည့်လို့ရအောင် shortcut တခုပေးထားပါတယ်။ အဲ့ဒါကတော့ A command ပဲဖြစ်ပါတယ်။ ဒါကိုစမ်းကြည့်ပြီးတော့ကျွန်တော်တို့ရဲ့ file ထဲကိုစာကြောင်းတွေထပ်ထည့်ကြည့်ကြစို့။

The quick brown fox jumped over the lazy dog. **It was cool** .

Line 2

Line 3

Line 4

Line 5

နောက်တကြိမ်ထပ်ပြီးတော့ insert mode ထဲကထွက်ဖို့အတွက် Esc key ကိုနှိပ်လိုက်ပါ။

ကျွန်တော်တို့မြင်ရတဲ့အတိုင်းပဲ A command ဟာပိုပြီးတော့အသုံးဝင်ပါတယ်။ ဘာလို့လဲဆိုတော့ သူက insert mode ထဲမဝင်ခင် cursor ကိုစာကြောင်းအဆုံးကိုရွှေ့ပေးလို့ပါပဲ။

Opening a Line

ကျွန်တော်တို့စာတွေထပ်ထည့်လို့ရတဲ့နောက်တနည်းကတော့ စာကြောင်းအသစ်ထပ်ယူတာပါပဲ။ အဲ့ဒါက စာကြောင်းနှစ်ခုကြားမှာစာကြောင်းအလွတ်တခုရလာစေပြီးတော့ insert mode ထဲကိုဝင်ပေးပါတယ်။ Table 12-2 မှာပြထားသလိုသူ့မှာကွဲလွဲချက် ၂ ခုရှိနေပါတယ်။

Table 12-2: Line Opening Keys

Command	Opens
o (အို အသေး)	လက်ရှိစာကြောင်းရဲ့အောက်
O (အို အကြီး)	လက်ရှိစာကြောင်းရဲ့အပေါ်

ဒါကိုကျွန်တော်တို့ကဒီလိုလက်တွေ့လုပ်လို့ရပါတယ်။ cursor ကို line 3 မှာထားလိုက်ပြီးတော့ o (အို အသေး) ကို နှိပ်လိုက်ပါ။

The quick brown fox jumped over the lazy dog. **It was cool** .

Line 2

Line 3

Line 4

Line 5

တတိယစာကြောင်းရဲ့အောက်မှာ line အသစ်တခုရသွားပြီးတော့ကျွန်တော်တို့က insert mode ထဲကိုဝင်သွားပါတယ်။ insert mode ထဲကထွက်ဖို့အတွက် Esc key ကိုနှိပ်ပါ။ ကျွန်တော်တို့လုပ်ထားတဲ့အပြောင်းအလဲတွေကို undo လုပ်ဖို့ u ကိုနှိပ်ပါ။ O (အို အကြီး) ကိုနှိပ်ပြီးတော့ cursor ရဲ့အပေါ်မှာလိုင်းတလိုင်းအသစ်ယူလိုက်ပါ။

The quick brown fox jumped over the lazy dog. **It was cool** .

Line 2

Line 3

Line 4

Line 5

insert mode ထဲကထွက်ဖို့အတွက် Esc key ကိုနှိပ်ပြီးတော့ ကျွန်တော်တို့လုပ်ထားတဲ့အပြောင်းအလဲတွေကို undo လုပ်ဖို့ u ကိုနှိပ်ပါ။

Deleting Text

ကျွန်တော်တို့မျှော်လင့်ထားတဲ့အတိုင်းပဲ vi အနေနဲ့စာတွေဖျက်တဲ့နည်းလမ်းမျိုးစုံကိုပေးထားပြီးတော့သူတို့အားလုံးမှာလည်း keystroke j ချက်ထဲက တချက်တော့ပါဝင်ကြပါတယ်။ ပထမဆုံးကတော့ X key ဖြစ်ပြီးတော့သူက cursor ရှိတဲ့နေရာက character ကိုဖျက်ပေးပါတယ်။ x ရဲ့အရှေ့မှာနံပါတ်တခုကိုထည့်ပေးပြီးတော့ character ဘယ်နှစ်လုံးဖျက်ချင်တယ်ဆိုတာကိုအတိအကျပြောပေးလို့ရပါတယ်။ D key ကတော့ပိုပြီးအထွေထွေသုံးဆန်ပါတယ်။ x လိုပဲသူ့ရဲ့အရှေ့မှာနံပါတ်တခုကိုထည့်ပေးပြီးတော့ character ဘယ်နှစ်လုံးဖျက်ချင်တယ်ဆိုတာကိုအတိအကျပြောပေးလို့ရပါတယ်။ နောက်တခုက d အနောက်မှာ ဖျက်မယ့် size ကို control လုပ်ပေးတဲ့ movement command တခုအမြဲတမ်းပါပါတယ်။ Table 12-3 မှာဥပမာတချို့ကို list ထုတ်ပြပေးထားပါတယ်။

ကျွန်တော်တို့စာတွေရဲ့ပထမလိုင်းက It ဆိုတဲ့စကားလုံးအပေါ်မှာ cursor ကိုထားလိုက်ပါ။ စာတွေအကုန်ပျက်သွားတဲ့အထိ x ကိုဆက်တိုက်နှိပ်လိုက်ပါ။ နောက်ပြီးရင်ဖျက်တားတာတွေအကုန် undo (ပြန်ပေါ်လာ) ဖြစ်တဲ့အထိ u ကိုဆက်တိုက်နှိပ်လိုက်ပါ။

Note: vi က undo တခါပဲလုပ်လို့ရပါတယ်။ vim ကတော့အကြိမ်များစွာလုပ်လို့ရပါတယ်။

Table 12-3: Text Deletion Commands

Command	Deletes
x	လက်ရှိ character ကိုဖျက်ပေးမယ် ။
3x	လက်ရှိ character နဲ့သူ့အနောက်က character ၂ ခုကိုဖျက်ပေးမယ် ။
dd	လက်ရှိ line တခုလုံးဖျက်ပေးမယ် ။
5dd	လက်ရှိ line နဲ့နောက်ထပ် ၄ line ဖျက်ပေးမယ် ။
dW	လက်ရှိ cursor ရှိတဲ့နေရာကနေ နောက်ကစကားလုံးတခုရဲ့အစနားအထိ ဖျက်ပေးမယ် ။
d\$	လက်ရှိ cursor ရှိတဲ့နေရာကနေ လက်ရှိ line ရဲ့အဆုံးအထိဖျက်ပေးမယ် ။
d0 (zero)	လက်ရှိ cursor ရှိတဲ့နေရာကနေ လက်ရှိ line ရဲ့အစအထိဖျက်ပေးမယ် ။
d^	လက်ရှိ cursor ရှိတဲ့နေရာကနေ လက်ရှိ line ထဲက non-whitespace character ရဲ့အစအထိဖျက်ပေးမယ် ။
dG	လက်ရှိ line ကနေ file ရဲ့အဆုံးအထိဖျက်ပေးမယ် ။
d20G	လက်ရှိ line ကနေ file ရဲ့ line ၂၀ မြောက်အထိဖျက်ပေးမယ် ။

ထပ်ပြီးတော့ဖျက်တာတွေလုပ်ကြည့်ကြစို့ ၊ အခုတကြိမ်မှာတော့ d command ကိုအသုံးပြုပါမယ် ။ ထပ်ပြီးတော့ cursor ကို It ဆိုတဲ့စကားလုံးဆီကိုရွှေ့လိုက်ပြီးတော့ dW လို့ရိုက်ပြီးဖျက်လိုက်ပါ ။

The quick brown fox jumped over the lazy dog. **was cool** .

Line 2

Line 3

Line 4

Line 5

d\$ လို့ရိုက်လိုက်ပြီးတော့ cursor ရှိနေတဲ့နေရာကနေ line အဆုံးအထိဖျက်လိုက်ပါ ။

The quick brown fox jumped over the lazy dog.

Line 2

Line 3

Line 4

Line 5

dG လို့ရိုက်လိုက်ပြီးတော့လက်ရှိ line ကနေ file အဆုံးအထိဖျက်လိုက်ပါ ။

~
~
~
~
~

u ခုခါရိုက်လိုက်ပြီးတော့ဖျက်ထားတာတွေကို undo လုပ်လိုက်ပါ။

Cutting, Copying, and Pasting Text

d command က text တွေကိုဖျက်ပေးရုံတင်မကပါဘူး cut လည်းလုပ်ပေးပါတယ်။ ကျွန်တော်တို့က d command ကိုသုံးလိုက်တဲ့အချိန်တိုင်းမှာ ဖျက်ထားတာတွေကို paste buffer (clipboard ကိုတွေးလိုက်ပါ) ထဲကို copy ကူးသွားပါတယ်။ အဲဒါကိုကျွန်တော်တို့ကနောက်ပိုင်းမှာ p command နဲ့ပြန်ခေါ်ပြီးတော့ buffer ထဲက content တွေကို cursor နောက်မှာ paste ချတာ ဒါမှမဟုတ် P command နဲ့ cursor ရဲ့ရှေ့မှာ paste ချတာလုပ်လို့ရပါတယ်။

y command ကိုသုံးပြီးတော့ text တွေကို " yank " (copy) လုပ်တာဟာ d command ကိုအသုံးပြုပြီးတော့ text တွေကို cut လုပ်တာနဲ့ဆင်တူပါတယ်။ Table 12-4 မှာ y command နဲ့အခြားmovement command တွေမျိုးစုံကိုပေါင်းပြီးတော့ဥပမာတချို့လုပ်ထားတာကို list ပြထားပေးပါတယ်။

Table12-4: Yanking Commands

Command	Copies
yy	လက်ရှိ line ကို copy လုပ်ပေးပါမယ်။
5yy	လက်ရှိ line နဲ့နောက်ထပ် ၄ line ကို copy လုပ်ပေးပါမယ်။
yW	လက်ရှိ cursor ရှိနေတဲ့နေရာကနေ နောက်ထပ်စကားလုံးရဲ့အစအထိ copy လုပ်ပေးပါမယ်။
y\$	လက်ရှိ cursor ရှိနေတဲ့နေရာကနေ လက်ရှိ line ရဲ့အဆုံးအထိ copy လုပ်ပေးပါမယ်။
y0 (zero)	လက်ရှိ cursor ရှိနေတဲ့နေရာကနေ လက်ရှိ line ရဲ့အစအထိ copy လုပ်ပေးပါမယ်။
y^	လက်ရှိ cursor ရှိနေတဲ့နေရာကနေ လက်ရှိ line ထဲကပထမဆုံး non-whitespace character အထိ copy လုပ်ပေးပါမယ်။
yG	လက်ရှိ line ကနေ file အဆုံးအထိ copy လုပ်ပေးပါမယ်။
y20G	လက်ရှိ line ကနေ file ထဲက line ၂၀ မြောက်အထိ copy လုပ်ပေးပါမယ်။

copy နဲ့ paste နည်းနည်းလုပ်ကြည့်ကြစို့။ text တွေထဲကပထမဦးဆုံး line မှာ cursor ထားပြီး yy နှိပ်လိုက်ပြီးတော့လက်ရှိတ line လုံးကို copy ကူးလိုက်ပါ။ ပြီးရင် cursor ကို နောက်ဆုံး line ရောက်အောင် G နှိပ်ပြီး ရွှေ့ပြီးတော့ p နှိပ်ပြီးတော့ copy ကူးလာတဲ့ line ကို လက်ရှိရောက်နေတဲ့ (နောက်ဆုံး line) မှာ paste ချလိုက်ပါ။

The quick brown fox jumped over the lazy dog. **It was cool.**

Line 2

Line 3

Line 4

Line 5

The quick brown fox jumped over the lazy dog. **It was cool.**

စောစောတုန်းကလိုပဲ u command ကကျွန်တော်တို့ပြုလုပ်ခဲ့တဲ့အပြောင်းအလဲတွေကို undo လုပ်ပေးမှာ ဖြစ်ပါတယ်။ file ရဲ့ နောက်ဆုံး line မှာ cursor ရှိနေစဉ်မှာ P (shift+p)ကိုနှိပ်လိုက်ပြီးတော့ လက်ရှိ line ရဲ့အပေါ်မှာ text တွေကို paste ချလိုက်ပါ။

The quick brown fox jumped over the lazy dog. **It was cool.**

Line 2

Line 3

Line 4

Line 5

The quick brown fox jumped over the lazy dog. **It was cool.**

Line 5

The quick brown fox jumped over the lazy dog. **It was cool.**

Table 12-4 ထဲက အခြားသော y command တွေကိုလည်းစမ်းကြည့်ပြီးတော့ p (p အသေး) နဲ့ P (p အကြီး) command တွေဘယ်လိုအလုပ်လုပ်သလဲဆိုတာကို သိအောင်လုပ်ပါ။ ခင်ဗျားလုပ်လို့ပြီးသွားတဲ့အခါမှာ file ကိုနုကို ကအတိုင်းပြန်ဖြစ်အောင်ပြန်လုပ်ထားပါ။ (p အသေး က cursor ရဲ့အောက်တကြောင်းမှာ paste ချပေးပြီးတော့ p အကြီးက cursor ရဲ့အပေါ်တကြောင်းမှာ paste ချပေးပါတယ်။)

Joining Lines

vi မှာ line တခုနဲ့ပက်သက်တဲ့သူ့ရဲ့အတွေးအခေါ်(စည်းကမ်း)တွေကတင်းကျပ်ပါတယ်။ ပုံမှန်အားဖြင့် တော့ line တခုရဲ့အဆုံးဆီကို cursor ရောက်သွားအောင်ရွှေ့ပြီးတော့ end-of-line character ကိုဖျက်ပြီးသူ့အောက်က line တခုနဲ့ join ဖို့ဆိုတာမဖြစ်နိုင်ပါဘူး။ ဒါကြောင့်မို့ vi က line တွေပေါင်းဖို့အတွက် တိကျတဲ့ command တခုကို ထည့်ပေးထားပါတယ်။ J ဖြစ်ပါတယ် (cursor ရွှေ့ဖို့ထားတဲ့ j နဲ့မမှားပါနဲ့။)

ကျွန်တော်တို့က cursor ကို line 3 မှာထားလိုက်ပြီးတော့ J command ကိုရိုက်လိုက်မယ်ဆိုရင် ဘာဖြစ်သွားမလဲဆိုတာကိုဒီမှာပြထားပါတယ်။

The quick brown fox jumped over the lazy dog. **It was cool.**

Line 2

Line 3 Line 4

Line 5

Search and Replace

vi မှာ ရှာဖွေမှုတွေအပေါ်မူတည်ပြီးတော့ cursor ရဲ့တည်နေရာကိုရွှေ့ပေးနိုင်တဲ့စွမ်းဆောင်ရည်ရှိပါတယ်။ သူကအဲဒါကို line တခုပေါ်မှာ သို့မဟုတ် file တခုလုံးပေါ်မှာလုပ်ပေးနိုင်ပါတယ်။ သူက user ဆီကနေ confirmation ရရင်သော်လည်းကောင်း၊ မရရင်သော်လည်းကောင်း text တွေကိုအစားထိုးပေးတာမျိုးလည်းလုပ်ပေးနိုင်ပါတယ်။

Searching Within a Line

f command က line တခုကိုရှာပေးပြီးတော့ cursor ကိုသူ့အနောက်က (text တွေထဲက) ကိုယ်သတ်မှတ်ပေးလိုက်တဲ့ character ဆီကိုရွှေ့ပေးပါတယ်။ ဥပမာ - fa ဆိုတဲ့ command ဟာ cursor ကိုလက်ရှိ line ထဲမှာ (ဗယ်ကနေညာကိုပဲ့လုပ်ပေးတယ်) ပထမဆုံးတွေ့ရမယ့် a ဆိုတဲ့စာလုံးနေရာဆီကိုရွှေ့ပေးပါတယ်။ line တခုအတွင်းမှာ character တခုကိုရှာပြီးတဲ့အခါမှာ semicolon (;) ကိုရိုက်လိုက်ပြီးတော့ ရှာဖွေမှုကိုနောက်ထပ်ပြုလုပ်နိုင်ပါတယ်။ (Ubuntu 20.04 ထဲက vi မှာတော့ ; ကအလုပ် မလုပ်ပါဘူး။ 0 (zero) ရိုက်ပြီး line ရဲ့ရှေ့ဆုံး ပြန်သွားပြီး f command နဲ့ ရှာချင်တဲ့ character တွဲရိုက်ပြီးအစကနေပြန်ရှာလို့ရပါတယ်။)

Searching the Entire File

စကားလုံးတလုံး သို့မဟုတ် စာပိုဒ်တပိုဒ်ရဲ့နောက်မှာ cursor ကိုရွှေ့သွားစေချင်ရင် / command ကိုအသုံးပြုပါတယ်။ အဲဒါက Chapter 3 မှာကျွန်တော်တို့လေ့လာခဲ့ပြီးဖြစ်တဲ့ less program မှာလုပ်ပေးတဲ့နည်းလမ်းနဲ့အတူတူပါပဲ။ ခင်ဗျားအနေနဲ့ / command ကိုရိုက်ထည့်လိုက်တဲ့အခါ screen ရဲ့အောက်ခြေနားမှာ forward slash (/) တခုပေါ်လာမှာဖြစ်ပါတယ်။ ပြီးရင်ကိုယ်ရှာချင်တဲ့ စကားလုံးတလုံး သို့မဟုတ် စာပိုဒ်တပိုဒ် ကိုရိုက်ထည့်လိုက်ပြီး Enter ရိုက်လိုက်ပါ။ cursor ကရှာဖွေပေးထားတဲ့စာသားနဲ့တူတဲ့နောက်ထပ်နေရာတခုဆီကိုရွှေ့သွားမှာဖြစ်ပါတယ်။ ရှာတဲ့အခါမှာ ပြီးခဲ့တဲ့ရှာဖွေမှုတုန်းကရှာခိုင်းထားတဲ့စကားလုံးနဲ့ပဲထပ်ရှာဖို့အတွက် n command ကိုသုံးပြီးတော့ ထပ်ခါထပ်ခါရှာဖွေနိုင်ပါတယ်။ ဒီမှာ ဥပမာ တခုလုပ်ပြထားပါတယ်။

The quick brown fox jumped over the lazy dog. It was cool.

Line 2

Line 3

Line 4

Line 5

cursor ကို file ရဲ့ပထမဆုံး line ပေါ်မှာတင်ထားလိုက်ပြီးတော့ အောက်ကစာကိုရိုက်ပြီးတော့ Enter နှိပ်လိုက်ပါ။ ။

/Line

cursor က line 2 ကိုရွှေ့သွားပါလိမ့်မယ်။ ။ ပြီးရင် n လို့ရိုက်လိုက်ပြီးတော့ cursor က line 3 ကိုရွှေ့သွားပါလိမ့်မယ်။ ။ n command ကိုထပ်ခါထပ်ခါရိုက်ခြင်းဖြင့် cursor က (ကိုယ်ရှာခိုင်းထားတဲ့စာနဲ့) တူတဲ့စာသားတွေမကုန်မချင်း file ရဲ့အောက်ဆုံးအထိကိုရွှေ့သွားမှာဖြစ်ပါတယ်။ ။ ကျွန်တော်တို့အနေနဲ့အခုအထိ စကားလုံးတလုံးသို့မဟုတ် စာပိုဒ်တိုဒ် ကိုပဲရှာနေပေမယ့် vi က regular expression သုံးတာကိုလည်းခွင့်ပြုပေးထားပါတယ်။ ။ ရှုပ်ထွေးတဲ့စာသားတွေကိုဖော်ပြရာမှာ အစွမ်းထက်တဲ့နည်းစနစ်တခုဖြစ်ပါတယ်။ ။ ကျွန်တော်တို့ Chapter 19 ကျမှ regular expression တွေအကြောင်းအချို့ကိုအသေးစိတ် ဆွေးနွေးသွားမှာဖြစ်ပါတယ်။ ။

Global Search and Replace

vi က ex command ကိုအသုံးပြုပြီးတော့ (စာတွေကို) ရှာဖွေတာနဲ့အစားထိုးတာတွေလုပ်ဆောင်ခြင်းကို (vi မှာတော့ substitution လို့ခေါ်တယ်) line အကွာအဝေးတခုအထိ သို့မဟုတ် တ file လုံးပေါ်မှာပြုလုပ်ပေးပါတယ်။ ။ တ file လုံးထဲကစာတွေကိုတကြောင်းချင်းစီလိုက်ပြောင်းဖို့အတွက်ဆိုရင်ကျွန်တော်တို့အနေနဲ့အောက်ပါ command ကိုရိုက်ထည့်မှာဖြစ်ပါတယ်။ ။

:%s/Line/line/g

ဒီ command ကိုသီးခြားအရာတွေအဖြစ်ခွဲထုတ်လိုက်ပြီးတော့တခုချင်းစီကဘာအလုပ်တွေလုပ်ပေးတယ်ဆိုတာကိုကြည့်ကြစို့။ ။ (Table 12-5 ကိုကြည့်ပါ)

Table12-5: An Example of Global Search-and-Replace Syntax

Item	Meaning
:	colon character (:) က ex command ကိုစတင်ပေးပါတယ်။ ။
%	line ဘယ်နှစ်ခု လုပ်ဆောင်မယ်ဆိုတာ(range)ကိုအတိအကျသတ်မှတ်ပေးပါတယ်။ ။ % ဟာ shortcut ဖြစ်ပြီးတော့ ပထမဦးဆုံး line ကနေ နောက်ဆုံး line အထိလို့အဓိပ္ပါယ်ရပါတယ်။ ။ နောက်တခုအနေနဲ့ကလည်း range ကို 1,5 ဆိုပြီးတော့ အတိအကျပြောလို့ရသလို (ကျွန်တော့် file မှာ line က ၅ လိုင်းပဲရှိပါတယ်။) ဒါမှမဟုတ် 1,\$ လို့ပြောလို့ရပါတယ်။ ။ အဓိပ္ပါယ်က file ရဲ့ line 1 ကနေနောက်ဆုံး line အထိလို့ပြောတာပါပဲ။ ။ line ရဲ့ range တွေကိုမထည့်ပဲချန်ထားခဲ့မယ်ဆိုရင် လက်ရှိ line ပေါ်မှာပဲလုပ်ပေးပါလိမ့်မယ်။ ။
s	ဘာလုပ်မယ်ဆိုတာကိုအတိအကျညွှန်ပြပါတယ်။ ။ အခုကိစ္စမှာတော့ substitution (ရှာတာနဲ့အစားထိုးတာ)
/Line/line	

gg	
----	--