



Linux Training for New Users of Cluster

Georgia Advanced Computing Resource Center

University of Georgia

Suchitra Pakala

pakala@uga.edu



Overview

- GACRC
- Linux Operating System
- Shell, Filesystem, and Common Commands
- Scripting and execution





Georgia Advanced Computing Resource Center

- We are a high-performance-computing (HPC) center at the UGA
- We provide to the UGA research and education community an advanced computing environment:
 - HPC computing and networking infrastructure located at the Boyd Data Center
 - Comprehensive collection of scientific, engineering and business applications
 - Consulting and training services
- <http://wiki.gacrc.uga.edu> (Wiki)
- <https://uga.teamdynamix.com/TDClient/Requests/ServiceCatalog?CategoryID=11593> (User Support)
- <http://gacrc.uga.edu> (Web)



- Introduction to Linux
- Connecting to a Linux machine



Linux Operating System

- Operating System (OS)
 - Software program
 - Enables hardware to communicate and operate with software
 - Manages all resources and applications
 - Memory, File system, Networking, I/O, etc.
 - Browser, Video player, etc.
- Most popular Operating Systems : Mac, Linux, Windows.



Linux Operating System

- About Linux OS
 - Developed in 1991 by Linus Torvalds
 - Open Source
 - Multi-user, Multi-tasking operating system
 - Most popular OS in the high performance computing community
 - Several distributions - Ubuntu, CentOS, Fedora, RedHat, etc.
- Why use Linux?
 - Free, Stable, Secure, Portable, Scalable



Linux Operating System

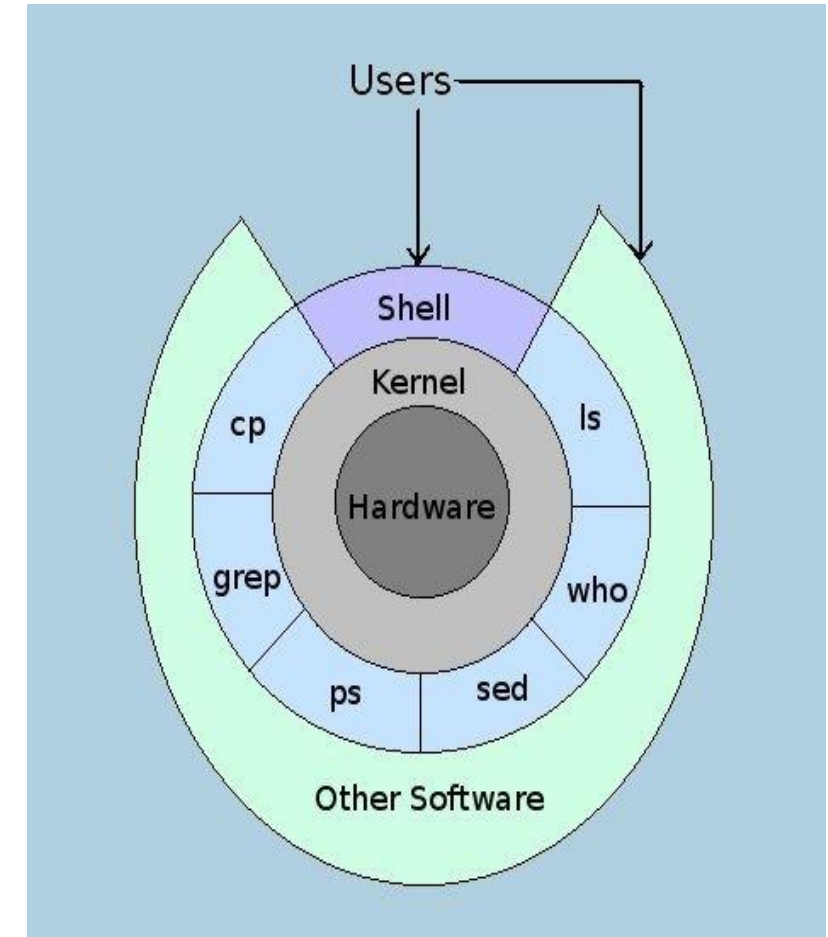
➤ Two major components of Linux:

➤ **Kernel**

- Core of the OS
- Schedules processes, and interfaces with hardware
- Manages resources – memory, I/O, etc



➤ **Shell**

- The shell is an interface between users and the kernel
- Command-line – Users can type commands
- Command interpreter – runs commands
- Programming environment – for scripting





Linux Shell

- “Shell” - **command line interpreter**
 - Interacts between the system and users
 - Reads commands from the keyboard
 - Executes commands
 - Displays the output
 - Provides the “environment”
 - Command-line completion
 - Auto-correction
 - TAB key - Auto-completion
 - Up  and down  arrow keys - command history
 - Several shells available
 - **Bash-shell** (bash) is the default one.



Connecting to Shell - on Mac/Linux



- Open a terminal and type: `ssh <UGAMyID>@sapelo2.gacrc.uga.edu`
- Enter your Password when prompted
- Note: The password entry will not show on the screen. Not even asterisks.

```
suchitra@Lenovo-Suchi ~  
$ ssh pakala@sapelo2.gacrc.uga.edu  
  
UGA DUO authentication is required for SSH/SCP access to  
GACRC systems.  
  
UGA DUO is a two-factor authentication service which  
requires a password (one factor) and a code, phone,  
or device (second factor) to successfully authenticate.  
  
If you are not enrolled in the UGA DUO service please  
visit the UGA DUO service self-service portal to enroll  
and configure or manage your DUO enabled devices.  
  
https://eits.uga.edu/access\_and\_security/infosec/tools/duo/portal/  
  
For additional help with UGA DUO authentication or to  
report an issue please visit:  
  
https://eits.uga.edu/access\_and\_security/infosec/tools/archpass/  
  
Password:  
Duo two-factor login for pakala  
  
Enter a passcode or select one of the following options:  
  
1. Duo Push to XXX-XXX-3898  
2. Phone call to XXX-XXX-3898  
3. SMS passcodes to XXX-XXX-3898  
  
Passcode or option (1-3): 1  
Success. Logging you in...
```

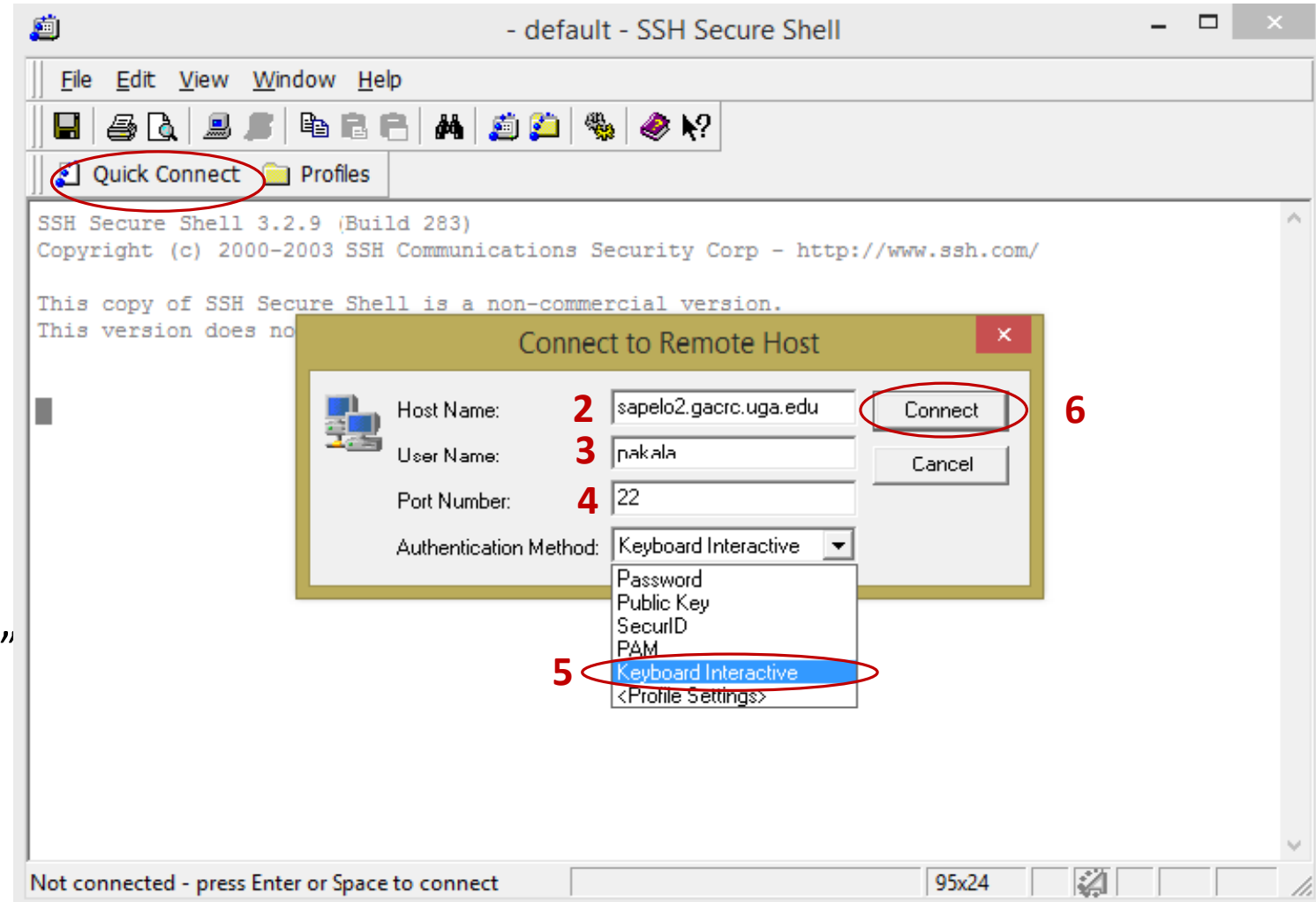


Connecting to Shell – on Windows

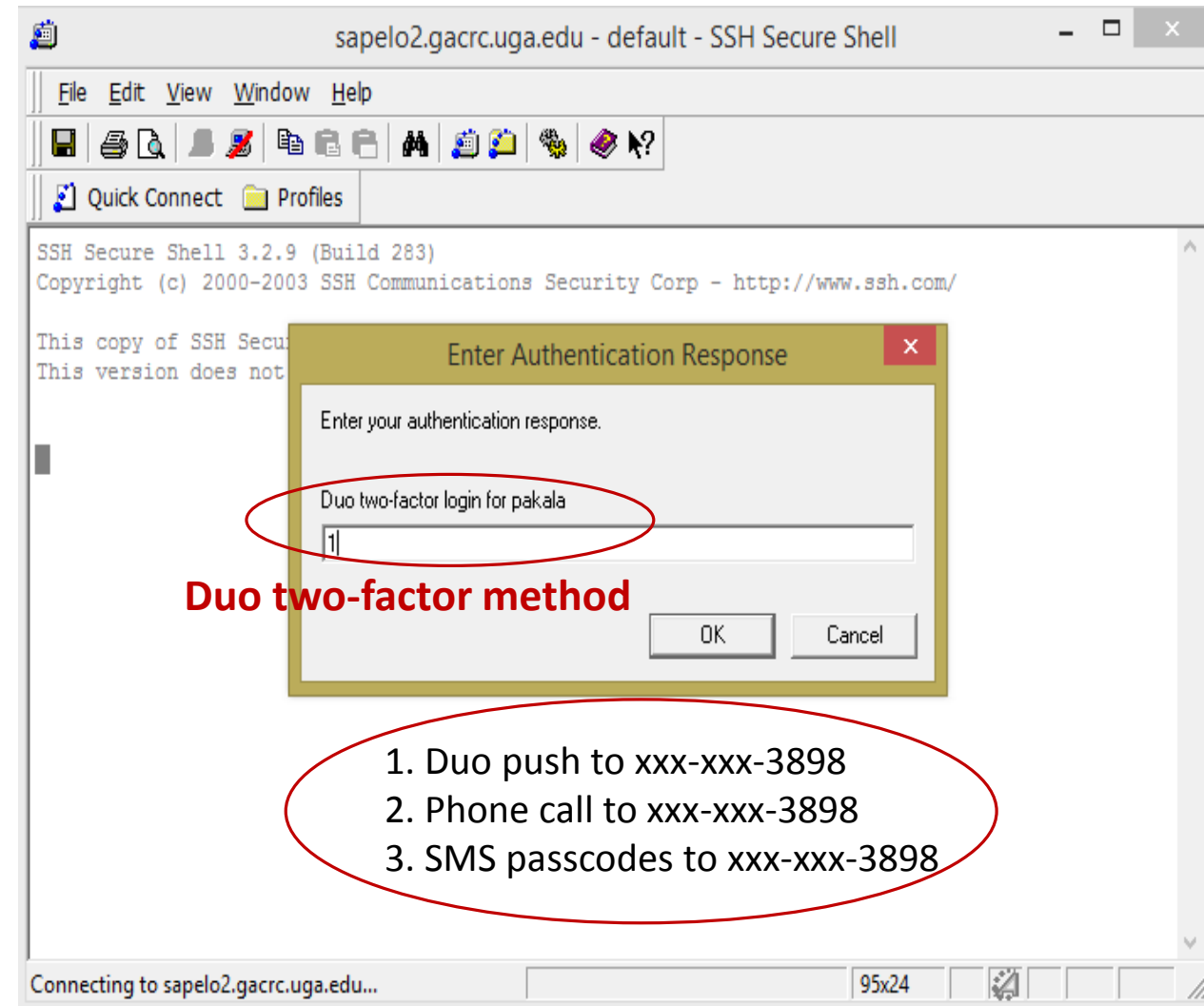
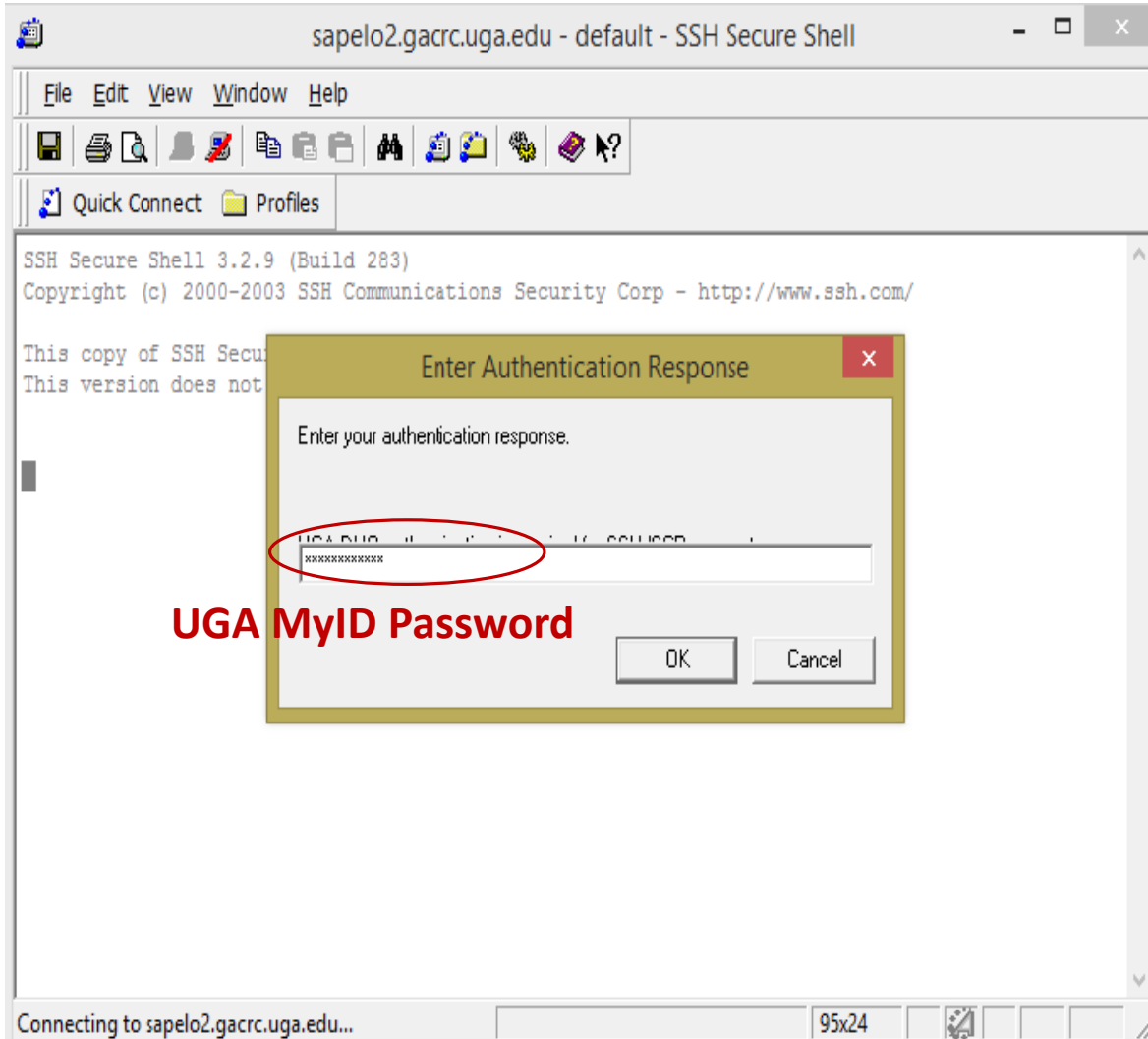


➤ Download SSH Secure Shell from http://eits.uga.edu/hardware_and_software/software/

1. Open the SSH Secure Shell and click on "Quick connect".
2. Hostname: **sapelo2.gacrc.uga.edu**
3. User Name: **your UGA MyID**
- 4. Port Number: 22
- 5. Authentication Method: Select:
Keyboard Interactive
6. Enter above information and click "Connect"
- Enter your password in the next pop up window and click "OK"



Connecting to Shell – on Windows

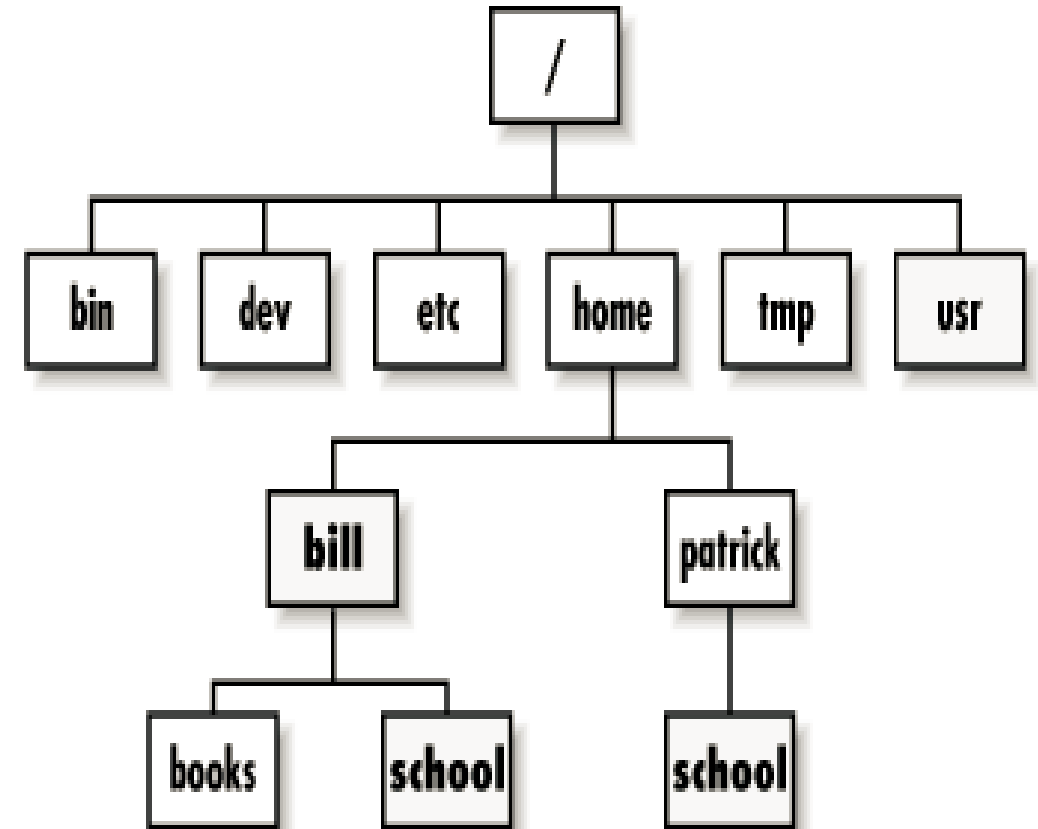


- Linux Directory Structure
- Navigation Commands



Understanding Linux Directory Structure

- 'upside down tree'
- **Root directory** ("/" forward slash)
- Organized inside root directory
- Create directories inside - **sub directories**
- Unique name in its containing directory



Relative Path vs Absolute Path

➤ **Relative path**

- Path to a file, relative to current location (present working directory)

```
$ pwd                                     ← Present working directory
/home/pakala/

$ ls
Suchi_Scripts    Blast

$ ls Blast                                  ← Relative Path
AF293            Escherichia_Coli_LF82.fasta
blast.sh        GCF_000002655.1_genomic.fna
```

➤ **Absolute or Full path**

- Path to a file, beginning at the root

```
$ ls /home/pakala/Blast/ ← Absolute path
AF293            Escherichia_Coli_LF82.fasta
blast.sh        GCF_000002655.1_genomic.fna
```



Change Directory (cd)

➤ **cd** :change your current working directory

```
pakala@uga-2f0f976:~ $ cd /usr/bin ← Move to bin subdir of usr dir
pakala@uga-2f0f976:~ $ cd .. ← Move up one directory
pakala@uga-2f0f976:~ $ cd ← Returns to home directory
pakala@uga-2f0f976:~ $ cd ~pakala ← Returns to home directory/pakala
pakala@uga-2f0f976:~ $ cd $HOME ← Environment Variable/home dir
```

➤ **pwd** :present working directory

```
pakala@uga-2f0f976:~ $ pwd
/home/pakala
```



List Directory (ls)



➤ ls

➤ lists files and directories that exist in the current location

➤ Note: we cannot differentiate between files and directories

```
pakala@uga-2f0f976:~ $ ls  
e_coli_data.fq  hello.sh  sample_script  sub.sh  Suchi_Scripts
```

➤ ls -l

➤ shows file permissions, owner of file, group, file size, modified date and time, and differentiates between file or directory name.

```
pakala@uga-2f0f976:~ $ ls -l  
total 584496  
-rw-r--r-- 1 pakala abclab 1610499990 Mar  6 09:46 e_coli_data.fq  
-rwxr----- 1 pakala abclab      136 Feb 21 15:22 sample_script  
-rw-r--r-- 1 pakala abclab      284 Mar  6 09:50 sub.sh  
drwxr-xr-x 2 pakala abclab         2 Feb 22 12:07 Suchi_Scripts
```





List Directory (ls)

➤ **ls -a**

- Lists hidden files. They start with ‘.’
- These are files containing profiles and other settings that should not be altered unless necessary, and hence are “hidden”

```
pakala@uga-2f0f976:~ $ ls -a
.          .bash_history.n609      .emacs          .mozilla
..         .bash_history.sapelo2-sub1 .emacs.d        .oracle_jre_usage
.bash_history .bash_history.sapelo2-sub2 .felix          sample_script
.bash_history.n201 .bash_logout          .fontconfig    .ssh
.bash_history.n204 .bash_profile         .gnome2        sub.sh
.bash_history.n206 .bashrc               hello.sh       Suchi_Scripts
.bash_history.n210 .beast                .java          .swp
.bash_history.n227 .cache                .ldaprc        .viminfo
.bash_history.n233 .config               .lmod.d
.bash_history.n234 e_coli_data.fq       .matlab
```



List Directory (ls)

➤ ls -lh

- shows sizes in human readable format

```
pakala@uga-2f0f976:~ $ ls -lh
total 571M
-rw-r--r-- 1 pakala gclab 1.5G Mar  6 09:46 e_coli_data.fq
-rwxr----- 1 pakala gclab  136 Feb 21 15:22 sample_script
-rw-r--r-- 1 pakala gclab  284 Mar  6 09:50 sub.sh
drwxr-xr-x 2 pakala gclab    2 Feb 22 12:07 Suchi_Scripts
```

➤ ls -lS

- Displays file size in order

```
pakala@uga-2f0f976:~ $ ls -lS
total 584496
-rw-r--r-- 1 pakala gclab 1610499990 Mar  6 09:46 e_coli_data.fq
-rw-r--r-- 1 pakala gclab    284 Mar  6 09:50 sub.sh
-rwxr----- 1 pakala gclab    136 Feb 21 15:22 sample_script
drwxr-xr-x 2 pakala gclab    2 Feb 22 12:07 Suchi_Scripts
```



- Files
- Permissions
- Creation, Deletion, Copy and Move
Commands





Files And Processes

➤ File

- Collection of data
- Location of a file – Path
- Can be created using text editors (nano, vi, etc)

➤ Process

- Any program that is run
- Unique process identifier - PID
- For example: “ps” command which lists all processes

```
pakala@uga-2f0f976:~ $ ps
  PID TTY          TIME CMD
 21505 pts/225    00:00:00 bash
 24908 pts/225    00:00:00 ps
```



Files And File Names

➤ File

- Basic unit of storage for data
- May contain any characters
- File names are always **case sensitive**
- You should **avoid spaces, quotes, and parenthesis**
- File names can be long, up to 255 characters

➤ Directory

- Special type of file
- Holds information about other files
- Present working directory (pwd)

```
pakala@uga-2f0f976:~ $ pwd  
/home/pakala
```



File Permissions

- Multi-user environment
- File permissions are used to protect users and system files.
- The types of permissions a file can have are:

Read Permissions	Write Permissions	Execute Permissions
r	w	x

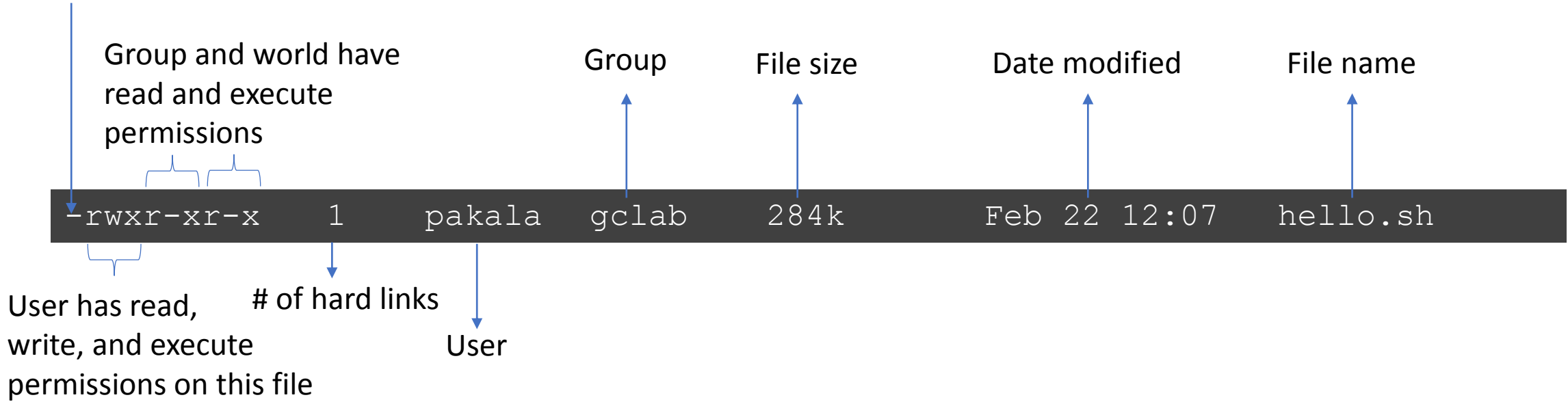
- Files and directories have three levels of permissions:
 - **User**
 - **Group**
 - **World**

User (owner)	Group	Others (everyone else)
rwX	rwX	rwX



File Permissions

File Type: - Regular file(d for Directory)



Changing File Permissions



- **chmod** command to change permissions of a file.
 - Symbolic mode:
 - Syntax: `chmod [references][operator][modes]`
 - References – “**u**” for user, “**g**” for group, “**o**” others
 - “**a**” for all three types
 - The operator – “**+**” to add and “**-**” to remove

```
>> Default settings when file was created:  
pakala@uga-2f0f976:~ $ ls -l  
-rw-r--r-- 1 pakala gclab 24 Feb 15 10:35 sample_script
```

```
>> Adding x(excute) permission for the user:  
$ chmod u+x sample_script  
-rwxr--r-- 1 pakala gclab 24 Feb 15 10:45 sample_script
```

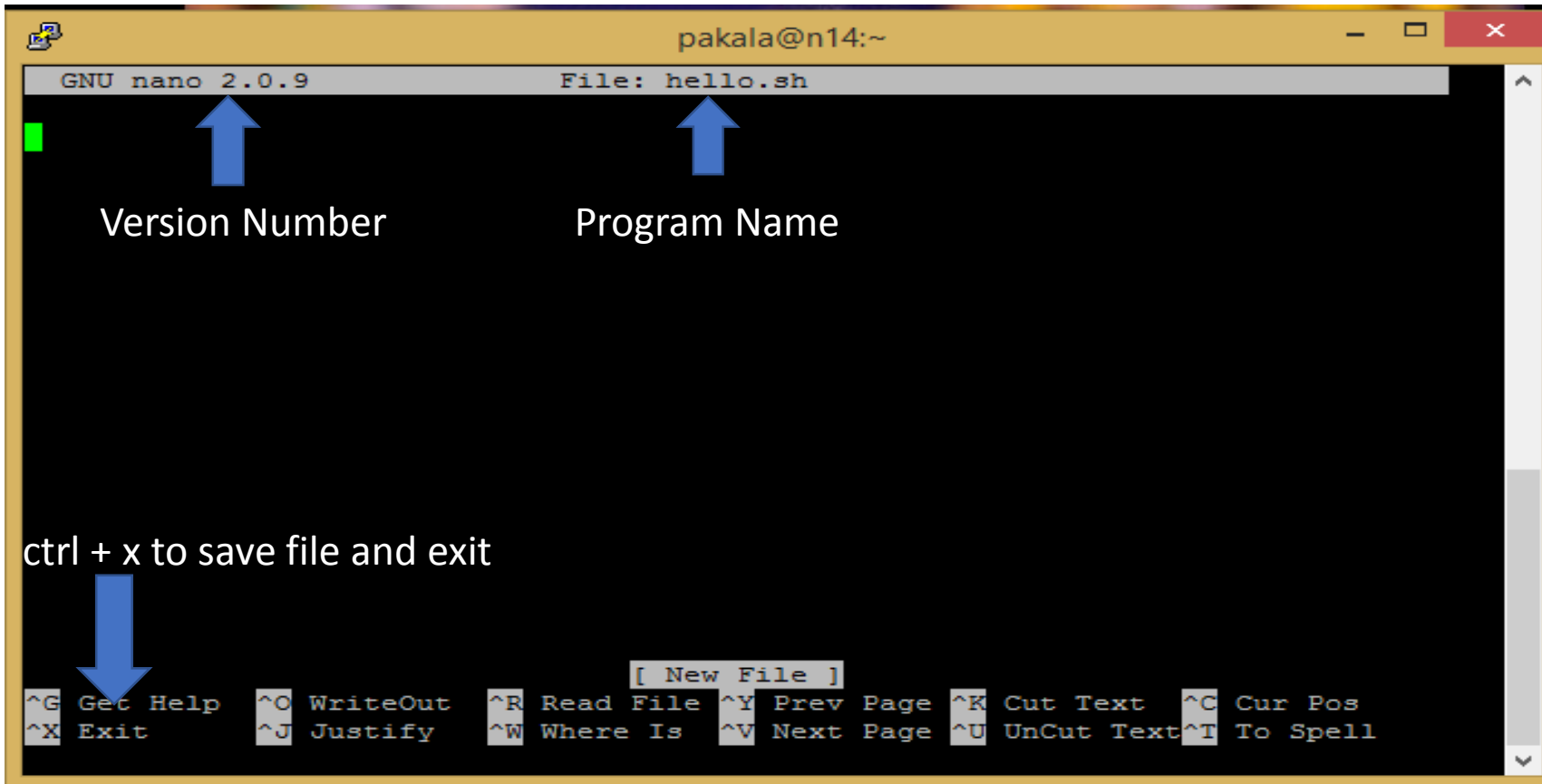
```
>> Removing r(read) permission for others:  
$ chmod o-r sample_script  
-rwxr----- 1 pakala gclab 24 Feb 15 10:50 sample_script
```



Creating and Editing Files

- Creating and editing files using a text editor
- The most widely used editors available on sapelo are vim, **nano**, etc

```
$ nano hello.sh
```



GNU nano 2.0.9 File: hello.sh

Version Number Program Name

ctrl + x to save file and exit

[New File]

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



Creating and Deleting Directories

- **mkdir** creates a directory

```
$ mkdir testdir
```

- Creating directories and subdirectories in one step

```
$ mkdir -p <dirname>/<subdirname>
```

- **rmdir** removes an empty directory

```
$ rmdir testdir
```

- Remove directories

```
$ rm -ri <directoryname> → Interactive Mode
```

- Removing Files

```
$ rm -i <filename> → Interactive Mode
```



Remove Files (rm)

- **rm** removes files

```
$rm -i /home/pakala/sample_script
```

- **Other options:**

option	description
	Remove (unlink) the FILE(s)
rm -f	ignore nonexistent files, never prompt
rm -i	prompt before every removal
rm -r, -R	remove directories and their contents recursively
rm -v	explain what is being done

- With the **-r** or **-R** option
 - **Removes entire directories recursively and permanently!!!**
- **rm -r *** option
 - Removes all of the files and subdirectories (**not recommended**)
- To remove an empty directory, use **rmdir**





Copy Files (cp)

- **cp** copies files or directories.
- To copy a file from /home/pakala/sample_script to /home/pakala/Suchi_Scripts

```
$ cp -i /home/pakala/sample_script /home/pakala/Suchi_Scripts
```

➤ Other Options:

option	description
cp -a	archive files
cp -f	force copy by removing the destination file if needed
cp -i	interactive - ask before overwrite
cp -n	no file overwrite
cp -R	recursive copy (including hidden files)
cp -u	update - copy when source is newer than destination
cp -v	verbose - print informative messages





Move Files (mv)

- **mv** moves a file to another location.
- For example, to move a file from `/lustre1/pakala/AF293.fs` to `/lustre1/pakala/Sample_Data`

```
$ mv -i AF293.fs /scratch/pakala/Sample_Data
```

- Can also be used to **rename** a file in the same directory.
- For example, to rename *myFile* to *myFileNew*:

```
$ mv myFile myFileNew
```

➤ Other options:

option	description
<code>mv -f</code>	force move by overwriting destination file without prompt
<code>mv -i</code>	interactive prompt before overwrite
<code>mv -u</code>	update - move when source is newer than destination
<code>mv -v</code>	verbose - print source and destination files
<code>man mv</code>	help manual



Summary of Common Linux Commands



- **cd** : Change your current working directory
- **pwd** : Print absolute path of your current working directory
- **ls** : List the files that exist in the current directory
- **mv** : Moves a file to another location.
- **cp** : Copies files or directories
- **mkdir** : Create a directory
- **rmdir** : Delete an empty directory
- **rm -r** : Delete a nonempty directory and its contents



More Linux Commands



- **file <filename>** : Display file type of file with name
- **cat textfile** : Throws content of text file on the screen
- **more <filename>** : Output the contents of a file
- **less <filename>** : Output the contents of a file
- **man <command>** : Read man pages command
- **dos2unix** : convert DOS/Windows file to Linux format
- **mac2unix**: convert mac file to Linux format
- **exit or logout**: leave the session



File Viewing

➤ **file** – determine the type of a file

```
pakala@uga-2f0f976:~ $ file Linux_Scripts/           ← directory
Linux_Scripts/: directory
pakala@uga-2f0f976:~ $ file e_coli_data.fg           ← ASCII text
e_coli_data.fg: ASCII text
pakala@uga-2f0f976:~ $ file hello.sh                 ← Shell Script
hello.sh: Bourne-Again shell script text executable
```

➤ **cat**

- cat is a standard Linux utility that concatenates
- Prints the content of a file to standard output

```
pakala@uga-2f0f976:~ $ cat temp.txt
Hello!!!!
Welcome to Linux world!
```



File Viewing

➤ more

- view text files - one page at a time, scroll down only
- spacebar to scroll down

```
pakala@uga-2f0f976:~ $ more testfile
```

➤ less

- view text files, one page at a time, scroll up and down
- space bar to scroll down
- key **b** to scroll up, Key **q** to quit

```
pakala@uga-2f0f976:~ $ less testfile
```



Manual Pages (man)

- Linux includes a built in manual for nearly all commands.
- Example: **man rm** (remove)

```
$ man rm
RM(1)                                User Commands                                RM(1)
```

NAME

rm - remove files or directories

SYNOPSIS

rm [OPTION]... FILE...

DESCRIPTION

This manual page documents the GNU version of rm. rm removes each specified file. By default, it does not remove directories.

If the `-I` or `--interactive=once` option is given, and there are more than three files or the `-r`, `-R`, or `--recursive` are given, then rm prompts the user for whether to proceed with the entire operation.

OPTIONS

Remove (unlink) the FILE(s).

`-f, --force`
ignore nonexistent files, never prompt

`-i` prompt before every removal

`-r, -R, --recursive`
remove directories and their contents recursively



File Conversion

➤ **dos2unix** : Convert DOS/Windows file to Linux format

- Example: dos2unix file1
- Removes DOS/Windows line endings in file1

```
$ dos2unix file1
```

➤ **mac2unix** : Convert Mac file to Linux format

- Example: mac2unix file1
- Removes Mac line endings in file1

```
$ mac2unix file2
```



➤ Shell Scripting

➤ Script Execution



Shell Scripting

- Shell Script - series of commands written in plain text file
- Why to write Shell Script?
 - To automate tasks that should be run daily
 - Build “pipelines” of commands and other programs to run
 - Serve as automatic documentation
 - Useful to create our own commands
 - Save lots of time



Example Script



```
#!/bin/bash

# rsync using variables

SOURCEDIR=/home/pakala/Linux_Scripts
DESTDIR=/scratch/pakala/backup_files/

rsync -avh $SOURCEDIR $DESTDIR

# compressing directory

compress=Linux_Scripts_$(date +%Y%m%d).tar.gz
tar -czf $compress /home/pakala

# Simple if/else statement, checking if the directory exists or not

directory="./Suchi_Scripts"

if [ -d $directory ]; then
    echo "Directory exists"
else
    echo "Directory does not exist"
fi
```



Variables in Shell

- What is a **“variable”**?
 - A character string to which we assign a value
 - Value could be a number, text, filename or any other type of data
 - Pointer to the actual data

- **There are two types of variables:**
 - System variables
 - User defined variables

- **System variables**
 - Created and maintained by Linux
 - Defined in CAPITAL LETTERS, user can reset their default values





System Variables

System Variable	Meaning	Example Value
HOME	User's home directory	/home/pakala
PATH	Path to binaries	/usr/bin:/sbin:/bin:/usr/sbin
PWD	Current working directory	/home/pakala
SHELL	Path to default shell	/bin/bash
USER	User who is currently logged in	pakala
TERM	Login terminal type of user	xterm
LD_LIBRARY_PATH	Shared library search path	

```
pakala@uga-2f0f976:~ $ echo $SHELL  
/bin/bash
```

```
pakala@uga-2f0f976:~ $ echo $HOME  
/home/pakala
```



User Defined Variables

- Created and maintained by user, defined in lower letters
- Syntax: **variable name=value**
- Rules for naming variable name
 - Don't put spaces on either side of the equal sign
 - Variables are **case sensitive**
 - Do not use ?,* etc, to name your variable names
- To print or access user defined variables
 - Syntax: **\$variable name**

```
$ no=10
$ echo $no           #will print 10
$ no =25             #no spaces on either side of equal sign
-bash: no: command not found
$ No=11
$ echo $No           #case sensitive, will print 11
```





Example Script – breaking it down

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



Location of shell to use

```
# rsync using variables
```



Comment line

```
SOURCEDIR=/home/pakala/Linux_Scripts  
DESTDIR=/scratch/pakala/backup_files/
```

```
rsync -avh $SOURCEDIR $DESTDIR
```



Actual command to run

```
# compressing directory
```

```
compress=Linux_Scripts_$(date +%Y%m%d).tar.gz  
tar -czf $compress /home/pakala
```

```
# Simple if/else statement, checking if the directory exists or not
```

```
directory="./Suchi_Scripts"
```

```
if [ -d $directory ]; then  
    echo "Directory exists"  
else  
    echo "Directory does not exists"  
fi
```



Run Shell Script



```
$ chmod u+x sample_script.sh ← Adding execute permission for User
```

```
$ ./sample_script.sh ← Running the script
```

```
sending incremental file list  
created directory /scratch/pakala/backup_files
```

```
Linux_Scripts/  
Linux_Scripts/.swp  
Linux_Scripts/car.sh  
Linux_Scripts/file2  
Linux_Scripts/file2.sh  
Linux_Scripts/first.sh  
Linux_Scripts/forloop.sh  
Linux_Scripts/sample  
Linux_Scripts/sampledata.sh  
Linux_Scripts/samplescript.sh  
Linux_Scripts/test1.sh  
Linux_Scripts/whileloop.sh  
Linux_Scripts/sample1/
```

```
sent 14.68K bytes  received 229 bytes  29.82K bytes/sec  
total size is 13.89K  speedup is 0.93  
tar: Removing leading `/' from member names  
tar: /home/pakala: file changed as we read it  
Directory exists
```



.bashrc

- .bashrc is a shell script that Bash runs whenever it is started interactively.
 - Think about all the startup programs that run when you start Windows
- It initializes an interactive shell session. You can put any command in this file that you would type at the command prompt
- A common thing to put in .bashrc are aliases that you want to always be available

```
# .bashrc
# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
# User specific aliases and functions
export PATH=/home/pakala/bin:$PATH

alias ls='ls --color=auto -l'
alias p="pwd"
```





THANK YOU 😊

