CORE-1501: Introduction to Shell Programming for Researchers

(or, Spellcasting for Science)

SNOLAB Common Orientiation for Research Experiences (SNOCORE)
Term: Winter 2026

Instructor: Stephen Sekula
Research Group Manager, SNOLAB
Professor of Physics, Queen's University
Adjunct Professor, Laurentian University

Queen's University

Laurentian University
Université Laurentienne

SNOLAB

Background art generated by Bing AI Image Creation

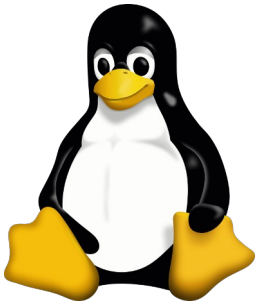
Overview

- What is GNU/Linux?
- How do I engage with GNU/Linux?
- Level-1 Shellcasting: files, folders, navigation, and actions
- Level-2 Shellcasting: pipes, string manipulation, and knowledge generation
- Next steps and outlook

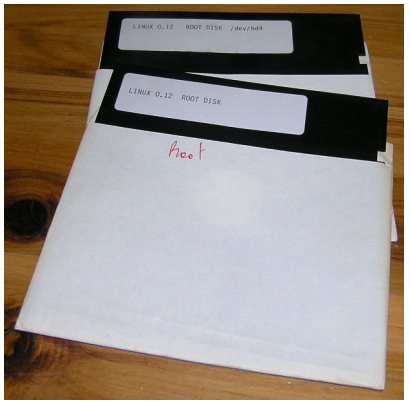
WHAT IS GNU/LINUX?

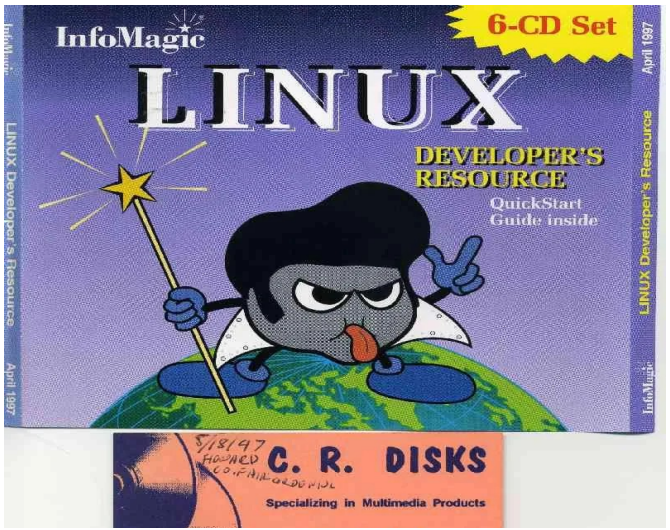
What is Linux?

- Linux is a “kernel” that connects computer hardware to an operating system (OS).
- It was developed by Linus Torvalds, is open-source, and was released originally in 1991.
- My personal journey into GNU/Linux began in 1997 when I switched to it from Windows 95 after that ?%!# OS crashed and cost me my umpteenth term paper. I installed Linux alongside Windows in the summer of 1997 and never looked back. SNOLAB is the first place I have been required to use Windows as my main OS since my university days. I’ve done everything on Linux at home and for work for 27 years.



Tux, the Linux mascot, created by Larry Ewing in 1996.





What is GNU?

Original logo by Etienne Suvasa

- First, what is UNIX?
  - UNIX is a collection of proprietary tools/commands for interacting with a computer from a terminal, and is what is meant by “the operating system (OS)” of a UNIX machine.
  - MacOS’s and iOS’s “Darwin” OS is built on FreeBSD UNIX, an open-source version of UNIX.
- GNU is Not UNIX! = GNU
  - Recursive acronym. Cute.
  - Released in 1983, initiated by Richard Stallman at the MIT Artificial Intelligence Laboratory. Managed by the Free Software Foundation (FSF).
  - A free, open-source collection of tools similar to UNIX, with familiar capabilities. GNU is really the open-source operating system. GNU and Linux together are usually referred to as “Linux”, but this makes some people unhappy.

HOW DO I INTERACT WITH GNU/LINUX?

If you're running Linux

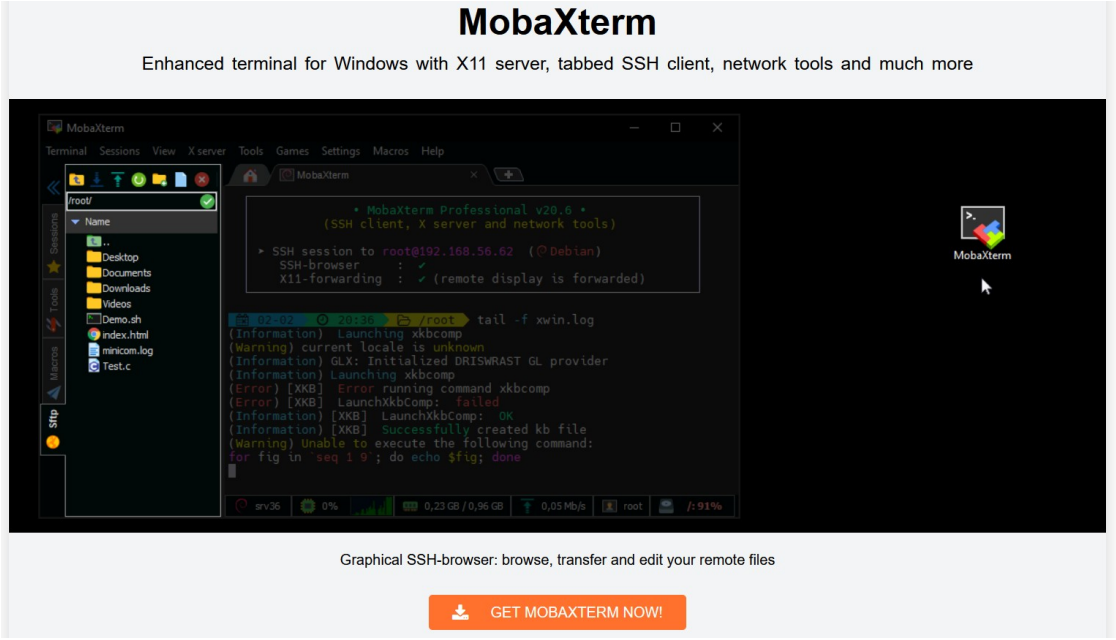
- Congratulations!
- You have made a wise OS choice.
- No further work is required. You can use a desktop graphical user interface (GUI, pronounced "goo-ee"), or run one of the many available terminal programs available.
- Examples: xterm, gnome-terminal, konsole, …

If you're NOT running Linux

- <<SAD TROMBONE>>
- These things happen.
- You have several options:
  - use a remote system
  - run a Linux virtual machine in your OS environment
  - Windows: install Linux Subsystem for Linux 2 (WSL 2) … a specific variant of the previous option.

Bonus: Windows + MobaXterm

- MobaXterm:
  - A one-stop shopping application that provides the basic tools needed to practice.



https://mobaxterm.mobatek.net/

Any OS: Run a Terminal

- A “Terminal” is from the ancient tongue (1950s) and refers to a physical device with the ability to enter commands and direct the actions of a computer.
- A “Terminal Emulator” is a software product that mimics one of those original terminal concepts.
- These days, people just say “terminal” to refer to emulators.
- Examples: Mac OS “Terminal” app or Windows “Powershell”,or “VSCode” on any OS, or third-party applications like Mobaxterm (Home Edition) on Windows.

A Digital Equipment Corporation (DEC) VT100, widely emulated in software

The Prompt

- A “prompt” or “command line” refers to a place where a human can enter a command for the OS.
- The environment in which that command is excecuted is provided by a “Shell”, which is an interpreter that processes the commands.
- Common shell environments in GNU/Linux are BASH, CSH, TCSH, and ZSH … but there are others! Shells provide a programming-language-like environment for a user to achieve higher-level actions.

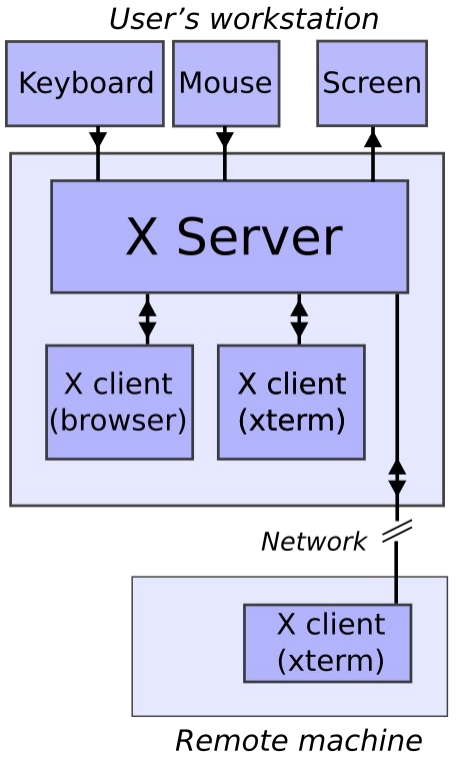
Lingo

A guide to crazy things I will say when giving instructions on typing in commands:

word/phrase	refers to...	word/phrase	refers to...
“dot”	.	“ampersand”	&
“tilde”	~	“star”	*
“dash”	-	“parenthesis”	(or)
“underscore”	_	“(square) bracket”	[or]
“caret”	^	“curly bracket”	{ or }
“at”	@	“space”	hit the spacebar!

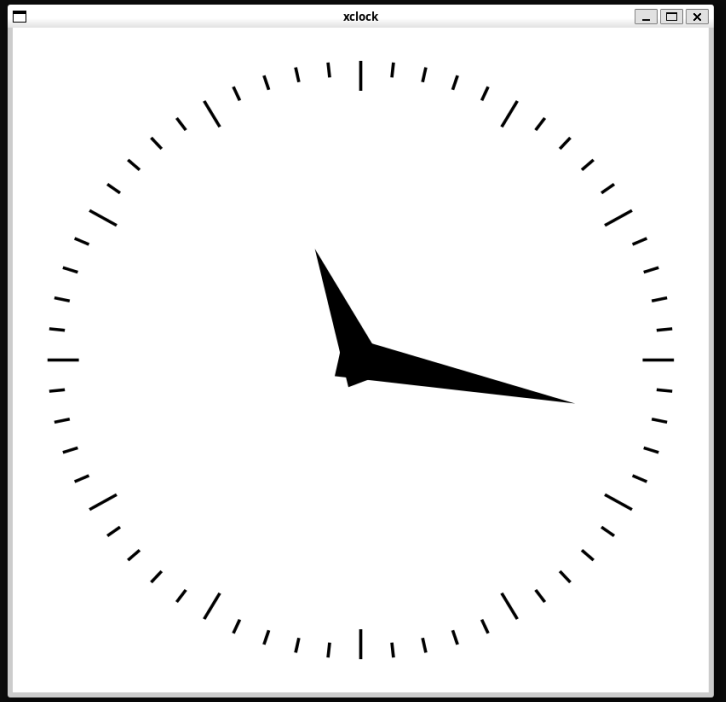
X Windows and Support

- How does LINUX do graphical desktop support?
  - Old way: a graphical system called “X Windows” or X11, or just “X”.
  - New way: Wayland (not fully adopted yet)
- In either case, the idea is that if you need to run a graphical application from a terminal, you need a system to “pop up” windows so you can interact with them.
- In Windows: you can run an X server using VcXsrv



VcXsrv: Example

```
ssekula@SNOLABTB-150: ~
PowerShell 7.4.5
PS C:\Windows\System32> Ubuntu
(base) ssekula@SNOLABTB-150:~$
(base) ssekula@SNOLABTB-150:~$
(base) ssekula@SNOLABTB-150:~$ xclock
Warning: Missing charsets in String to FontSet conversion
```



Here, I use Powershell to launch WSL2, specifically an Ubuntu Linux environment, and run the old "xclock" program. VcXsrv lets Ubuntu pop up a clock window from a Powershell session.

Getting to a Remote Linux System

If you don't have Linux handy, no problem! SNOLAB's Nearline Computing System runs a suite of GNU/Linux tools based on CentOS 7, which is also the baseline choice of the Digital Research Alliance of Canada (DRAC) national computing resource.

We can open a terminal program and use the Secure Shell, or SSH, command to connect from our terminal to a remote terminal session on Nearline. For this you need to know the short version of your SNOLAB user name (e.g., mine is "ssekula").

```
# SSH into Nearline from a prompt in your terminal program on your local machine
> ssh ssekula@nearline-login.computing.snolab.ca

# The first time your connect to a new machine, SSH wisely asks if you really meant to
# do this. Make sure you typed the name of the machine correctly and say "Y" to proceed.

# Enter your SNOLAB password when prompted. You should see something like this:
Last login: Thu Sep  5 15:23:24 2024 from 172.31.184.183
================================================================================
Welcome to the SnoLab Internal Compute cluster!

================================================================================

[ssekula@nearline-login ~]$
```

LEVEL-1 SHELLCASTING:
FILES, FOLDERS, NAVIGATION, AND ACTIONS

The File System

A standard file system at its top most level looks like this on LINUX:

```
> ls /

bin
boot
dev
etc
home
init
lib
lib32
lib64
libx32
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

"/home" is where all system user personal files will be located

"/root" is where the main administrator account ("root account") files are stored

"/tmp": Volatile scratch space, but often essential to OS function … avoid filling it!

"/usr": Non-essential (to the OS) libraries and executable programs are installed here by default

Shellcasting 101: (Forward) Slash

"/" is the standard separating character between directories/folders in a file system. Used alone (e.g., "cd /"), "/" refers to the root of the file system – the top-most folder (the one folder that holds them all)

Navigating the File System

```
# list the contents of a folder or get details of a file or folder
> ls

# list all files, including hidden ones (beginning with a “.”), and give details!
> ls -laF

# Give me details about a very specific file
> ls -l <FILENAME>
```

```
# change directories on the file system
> cd <DIRECTORY>

# go up one level in the folder/directory hierarchy
> cd ..

# take me back to the last directory I was in!
> cd -

# Take me back to my home directory!
> cd ~

# WHERE AM I?! (present working directory)
> pwd
```

Shellcasting 101: Auto-Complete

Type the first few characters of a file or folder name and hit the <TAB> key. Most shell environments will attempt to auto-complete the name, speeding typing.

It's time for a

COFFEE BRAK!

BREAK!

LET'S TAKE A 5-MINUTE BREAK.

Looking in/at files (1)

```
# list the contents of a folder or get details of a file or folder
> cat <FILENAME>

# "cat" is short for "concatenate" - show me the contents of a bunch of files!
> cat <FILENAME1> <FILENAME2> … <FILENAME N>

# SHOW ME ALL THE FILES! (dangerous)
> cat *
```

```
# interactively look at a file instead of dumping all of it to the screen
# at once
> more <FILENAME>

# Hey! I need to do more than scroll forward (spacebar or down arrow) or
# jump backward ("b" key or up arrow). How do I have a better interactive
# experience?
> less <FILENAME>

# The joke? "less is more". Bah-dum-bum. IN less, you can use arrow keys to
# move line-by-line in the file, and spacebar and backspace to
page-down/page-up, etc.
```

Shellcasting 101: Wildcard

"*" is a powerful and dangerous character. Shells interpret it to mean "any file or folder name made from any character" ... so it matches EVERYTHING. Use it wisely.

Looking in/at files (2)

```
# search for specific text in a file, print all matching lines
> grep "astroparticle" <FILENAME>

# search for specific text, but insensitive to letter case
> grep -i "AstroParticle" <FILENAME>

# Just tell me how many lines match with this word or phrase
> grep -c "astroparticle physics" <FILENAME>

# Find lines that have any of the following words and the egrep command
> egrep "astro|neutron|neutrino|project" <FILENAME>
```

What dark magic is this? "Regular expressions"! These spells are best understood at Level-2.

Shellcasting 201: Regular Expressions

A system of symbols and patterns of symbols that permits searching and matching, including replacement, in text. AKA "regex" or "regexp".

Editing Files

```
# fully interactive file editors are a bit of a religion in GNU/Linux. Wars have
# been fought and never won between the communities that like EMACS and VI (or VIM).
# Those two communities are united only in both agreeing that the "pico" editor is for
# suckers. :-)
> emacs -nw <FILENAME>

# This starts emacs in "no window" mode (do everything in the terminal).

> vim <FILENAME>

# I won't say any more about these. On your own time, explore the rich
# language of "key sequences" for each that are required to save, exit, open
# additional files, etc. Command-line editors are their own subject. Ask your
# supervisor for recommendations/preferences. Get a cheat sheet!

# I use VSCode for serious work locally and remotely (it has the ability to
# connect to a remote system via SSH and make it look as if you are working
# locally while you are also editing remotely).
```

Copy and Move Files

```
# copy a file intact to another folder
> cp <FILENAME> <PATH>/<TO>/<FINAL>/<FOLDER/

# copy a file to another name
> cp <FILE1> <FILE2>

# move a file to another folder or filename (equivalent to copy and then delete)
> mv <FILE1> <PATH>/<TO>/<FILE2>
```

```
# Use wildcards to copy files matching certain conditions to another location
> cp *.dat <PATH>/<TO>/<DESTINATION>

# (the above matches all files with the extension ".dat" in the current folder)

# Move all folder contents (including files AND folders) to another location
> mv * <PATH>/<TO>/<DESTINATION>
```

Move files between systems

```
# Securely copy a single file from the current system to a remote networked system
> scp <FILENAME> <USERNAME>@<REMOTE_SYSTEM>:/<PATH>/<TO>/<DESTINATION>/

# EXAMPLE: copy file to my home directory on Nearline
> scp analysis.dat ssekula@nearline-login.computing.snolab.ca:.

# "." means "here", and the default location is your remote home directory
```

```
# Send a whole bunch of files, tell me the progress, and allow me to pick up where I
# left off if transfer fails (rsync!)
> rsync -av *.dat <USERNAME>@<REMOTE_SYSTEM>:/<PATH>/<TO>/<DESTINATION>/

# If the transfer is interrupted, rerun the above and the process picks up where it
# left off after first checking the original progress.
```

LEVEL-2 SHELLCASTING:
PIPES, STRINGS, AND KNOWLEDGE GENERATION

Pipes and Redirects

Output from one GNU/Shell command can be “piped” as input to another GNU/Shell command. This is accomplished by joining commands together with the “|” symbol.

```
# previous example: search for specific text in a file, print all matching lines
> grep “astroparticle” <FILENAME>

# pipe-based version of same goal
> cat <FILENAME> | grep “astroparticle”

# Seems clunkier, but you can sequence as many of these as you like to accomplish as task
> cat <FILENAME> | grep “astroparticle” | grep -v “dark matter”

# “grep -v” vetoes any lines that match the text
```

You can redirect output to a file instead of to screen, if you need to save it for later:

```
# Match all lines with “astroparticle” and save lines to a new file
> grep “astroparticle” <FILENAME> > astroparticle_text.txt
> less astroparticle_text.txt # interactively look at the file
```

Regular Expressions

```
# Match all lines that contain the word "dark" followed
# by "sector", including any line with any other characters
# (including none at all) between these words
> egrep "dark.*sector" <FILENAME>

# (egrep implements full extended regular expression syntax,
#  achieving the same results as "grep -e")

# Match lines containing either "dark fermion" or "dark boson"
> egrep "dark (fermion|boson)" <FILENAME>

# Match lines containing either "Photomultiplier Tube" or "PMT"
> egrep "P(hotomultiplier|M)( ){0,1}T" <FILENAME>

# Will the above also match "PM T"? What about "PM  T"?
```

Shellcasting 201: Regular Expression Syntax

.	Match any character, including a space.
[ ]	Match a range of characters, e.g. [a-z] or [a-zA-Z] or [0-9]
*	Match the previous character or group 0 or more times
{m,n}	Match the previous character or group at least m and at more n times.
()	Group characters inside the parentheses into a pattern.
^	Match beginning of line
$	Match end of line

Editing Text Using Regexp

```
# Use Stream Editor (sed) to find all instances of "dark fermion" and replace "fermion"
# with "boson"
> cat <FILENAME> | sed -e "s/dark fermion/dark boson/g"

# Reorder the phase "dark is matter" to "matter is dark"
> cat <FILENAME> | sed -e "s/\(dark\) is \(matter\)/\2 is \1/"

# Some numbers are out-of-order in a comma-separated-value file. Reorder each line.
> cat data.csv
1,2,3
4,5,6
7,8,9

> cat data.csv | sed -e "s/\([0-9]\+\),\([0-9]\+\),\([0-9]\+\)/\3,\2,\1/"
# "+" (entered as "\+" in sed) means "match 1 or more of the previous group or
# character"
```

Simple Analysis

```
# Do some simple arithmetic on the command line using "bc" (an arbitrary precision
# numeric processing language)
> echo "2 + 5 + 8" | bc -l
15

# Compute the average of those three numbers
> echo "(2 + 5 + 8)/3" | bc -l
5.00000000000000000000

# Use AWK, a language designed for text processing, to compute the mean of the second
# column of numbers. In AWK, "NR" is the number of rows processed and is provided
# internally by the awk program. The "cut" GNU tool will be used to select the
# second column from the CSV file, using the comma as the delimiter.
> cat data.csv | cut -d "," -f 2 | awk "{SUM += $1} END {print SUM/NR}"
5

# Compute the mean and standard deviation of the second column
> cat data.csv | cut -d "," -f 2 | awk "{SUM += $1; SUMSQ += ($1)^2} END {print SUM/NR,sqrt((SUMSQ - SUM^2/NR)/(NR-1))}"
5 3
```

NEXT STEPS AND OUTLOOK

Next Steps

- Read more about each command and find others.
  - https://www.gnu.org/software/
- Get a GNU/Linux/Shell cheat sheet (c.f. https://cheatography.com/tag/linux/)
- Identify an aspect of your project where you can practice these commands and approaches simply by doing work you already need to do. (goal-based learning)
- Did you have trouble connecting to Nearline? Submit and IT ticket (email support@snolab.zendesk.com and tell them what you did and what error(s) you saw)
- Having problems with your computer and need help getting the right tools/environment installed? Again, request IT for help and also let your supervisor know … they can help too!

APPENDIX