Computing for Economists

Introduction to Unix and the Command Line

Toren Fronsdal and Ruby Zhang

Harvard University

Table of Contents

Introduction

Motivation

While the graphical user interface (GUI) is certainly easier to work with for many tasks, the command line interface (CLI) provides many advantages:

- **Efficiency**: can be faster than using a GUI for many tasks
- **Automation**: automate repetitive tasks
- **Reproducibility**: scripting allows operations to be repeated and run on different systems
- **Ubiquity and Uniformity**: can work with the same CLI across many systems, including on remote servers

Examples in the research process where the CLI is useful:

- Working with a version control system
- Executing a make script to run a data pipeline
- Controlling permissions for data access
- Accessing remote servers

The Unix Operating System

Operating Systems

Two main families of operating systems:

- Unix and Unix-like family
- Windows family

Today we will focus on working with the command line with a Unix or Unix-like OS (e.g., macOS).

If you have a Windows, you can access Unix-based command-line tools using Cygwin or Windows Subsystem for Linux.

Unix

Unix is a multitasking, multi-user operating system known for its stability, scalability, and portability.

The development of **Unix** began in 1969 at Bell Labs.

- Initially designed for mainframe computers at AT&T
- Unix-like OS (e.g., Linux) now widely used in many computing devices
    - Computers (macOS), phones (iOS and Android), televisions (Samsung and LG Smart TVs), cars (Toyota, Tesla, Mercedes-Benz), data centers, web servers, all 500 largest supercomputers, etc.

Unix Philosophy

The Unix philosophy emphasizes simplicity, modularity, and composability, with the goal of creating software that is easy to understand, flexible, and efficient.

- Each program does one thing, and does it well
- Expect output of one program to be an input to another

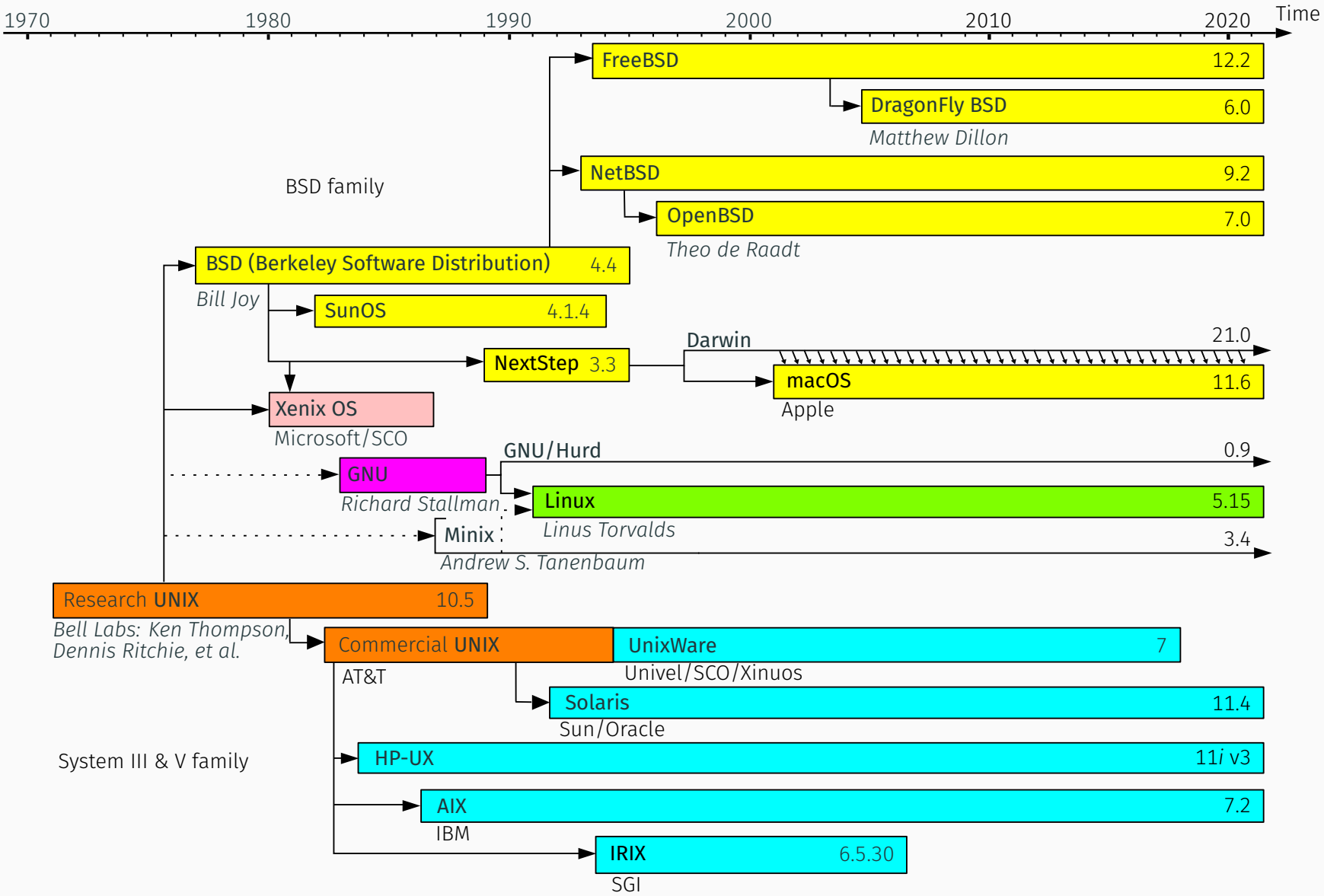
> *"Many UNIX programs do quite trivial things in isolation, but, combined with other programs, become general and useful tools."* *(The Unix Programming Environment)*

Linux

Linux is an open-source operating system kernel based on Unix that was originally developed by Linus Torvalds in 1991. It has become the most widely used OS.

- Linux kernel is responsible for allocating the computer's resources and scheduling user jobs
- Portable: can run on many different machines (over 95% of Linux is written in C)
- Most servers / computing clusters run Linux (e.g., Harvard FASRC)
- Popular Linux distributions include *`Ubuntu`*, *`Fedora`*, and *`Debian`*

Unix Family Tree



Source: Wikipedia

The Command Line

Unix Shell

The **Unix shell** is a command-line interface (CLI) program that provides users with an interactive way to communicate with the Unix operating system.

Common shells:

- *sh* (Bourne Shell)
- *bash* (Bourne Again Shell)
- *zsh* (Z Shell)

In 2019, Apple changed the default shell for macOS from *bash* to *zsh*.

- Easy to change the default shell
- Access shell via the **terminal**

Navigate the File System

Print working directory

```
$ pwd
/Users/torenfronsdal
```

List contents of a directory

```
$ ls
Applications        Dropbox
Desktop             Library
Documents           Public
```

Navigate to a different directory

```
$ cd Desktop
```

Shortcuts: `.` navigates to the “current directory”, `..` to the “parent directory”, and ~ to the “home directory”.

Create and Delete Files and Directories

Create a new directory

```
$ mkdir research-papers
```

Create a file

```
$ touch README.txt
```

Copy a file or folder

```
$ cp README.txt README_copy.txt
```

Move a file or folder

```
$ mv README_copy.txt ../README_copy.txt
```

Delete a file

```
$ rm ../README_copy.txt
```

Delete a folder

```
$ rmdir ../research-papers
```

Print, Search, and Compare Files

Display contents of a text file

```
$ cat README.txt
This directory contains the papers I will study this
    semester.
Lerner and Tirole, JEP 2005
```

Search for certain text within files

```
$ grep 'JEP' README.txt
Lerner and Tirole, JEP 2005
```

Find files with a certain name

```
$ find . -name "*.txt"
README.txt
README_copy.txt
```

Standard Wildcards

Standard wildcards (globbing patterns) perform actions on more than one file at a time, or to find part of a phrase in a text file.

Wildcard	Description
?	Matches one position: *x?* matches *x1* but not *x10*.
***	Matches multiple positions: *x** matches *x1* and *x10*.
[]	Specify a range: *[1-3]* returns *1 2 3*.
{}	Perform expansion of all values. Use multiple for Cartesian product: *{1,2}{3,4}* returns *13 14 23 24*.
[!]	All matches excluding characters in brackets: *x[!1]* matches *x10* but not *x1*.

```
$ ls *_data_[2022-2024].{csv, tsv}
singapore_data_2022.csv     singapore_data_2023.csv     singapore_data_2024.tsv
malaysia_data_2022.csv      malaysia_data_2023.csv      malaysia_data_2024.csv
```

For more advanced pattern matching / text search, use **regular expressions** (regex).

Command Options

Commands take the form:

```
$ command [arg1] [arg2] ... [argn]
```

An **option** is an argument that modifies the effects of a command.

- Long options are given by two hyphens and a word (e.g., *--help*)
- Short options are given by a hyphen and letter (e.g., *-h*)

For example, *ls -a* prints all files and directories, including hidden files:

```
$ ls -a
.bash_history   Applications    Dropbox
.bashrc         Desktop         Library
.gitconfig      Documents       Public
```

Permissions

Three actions you can specify permissions for: *r*ead, *w*rite, and e*x*ecute.

Users to grant permissions to:

- *a* all users
- *u* the owner user
- *g* the owner group
- *o* others (neither *u*, nor *g*)

chmod stands for change mode, and allows you to change the permissions:

```
chmod {a,u,g,o} {+,-} {r,w,x} files
```

For example, *chmod a+r README.txt* allows *README.txt* to be read by all.

Input/Output Redirection

Redirecting standard input:

```
command [arguments] < filename
```

Redirecting standard output:

```
command [arguments] > filename
```

This overwrites any file named *filename*. To append the standard output to a file, use `>>`.

Piping:

```
command_a [arguments] | command_b [arguments]
```

Equivalent to redirecting standard output of *command_a* to a file and then using that file as standard input to *command_b*.

Command-Line-Based Editors

Vim, **Emacs**, and **Nano** are text editors commonly used in Unix-like operating systems. They operate entirely within a command-line interface (Emacs also has a GUI).

Vim is the most popular command-line editor.

- Highly configurable and allows for many customizable key bindings and plugins
- Steep learning curve

Command-line text editors have declined in popularity with the rise of GUI text editors like Visual Studio Code, Sublime Text, and Atom.

Shell Scripting

Scripting

Shell scripts are plain text files with the extension *.sh*. Unix commands can be written in shell scripts the same way they would be written in *bash*, allowing for repeatable execution.

Scripting

Shell scripts are plain text files with the extension *`.sh`*. Unix commands can be written in shell scripts the same way they would be written in *`bash`*, allowing for repeatable execution.

- Shell scripts can use *control flow* commands (*`if`* statements, *`for`* and *`while`* loops, etc.) in addition to ordinary commands
- Can also define *functions*
- Accept *arguments* provided at the time of execution that can be used within the script
  - *`$0`* retrieves the script name
  - *`$1-$9`* retrieves the arguments passed to the script (*`${i}`* for *`i`* > 9)
  - *`$@`* retrieves all arguments

Scripting

Shell scripts are plain text files with the extension `.sh`. Unix commands can be written in shell scripts the same way they would be written in `bash`, allowing for repeatable execution.

- Shell scripts can use *control flow* commands (`if` statements, `for` and `while` loops, etc.) in addition to ordinary commands
- Can also define *functions*
- Accept *arguments* provided at the time of execution that can be used within the script
  - `$0` retrieves the script name
  - `$1-$9` retrieves the arguments passed to the script (`${i}` for `i` > 9)
  - `$@` retrieves all arguments

`.bashrc` is a shell script that the `bash` shell runs whenever it is started interactively.

- Add variables, functions, prompts, etc. to `.bashrc` to ensure they are available every time you use the shell

Scripting

.bashrc is a shell script that the *bash* shell runs whenever it is started interactively.

- Add variables, functions, prompts, etc. to *.bashrc* to ensure they are available every time you use the shell

```
alias la='ls -a' # shortcut to show hidden files
alias rm='rm -i' # -i prompts user before deletion
alias ..='cd ..' # shortcut to move to parent directory

# function to change directory and list files
cdl() {
    cd $1; ls
}
```

Then we can run

```
$ source .bashrc
$ la
.bash_history   Applications    Dropbox
.bashrc         Desktop         Library
.gitconfig      Documents       Public
$ cdl Desktop
courses
personal
research
$ ..
$ pwd
/Users/torenfronsdal
```

Scripting

Shell scripts are plain text files with the extension *.sh*. Unix commands can be written in shell scripts the same way they would be written in *bash*, allowing for repeatable execution.

We can write a script called *append_signature.sh*:

```
#!/bin/bash

echo "Starting program at $(date)"
# "$#" retrieves the number of arguments
echo "Running program $0 with $# files"
for file in "$@"; do
    echo "- Toren Fronsdal" >> "$file"
done
```

Scripting

Shell scripts are plain text files with the extension *.sh*. Unix commands can be written in shell scripts the same way they would be written in *bash*, allowing for repeatable execution.

We can write a script called *append_signature.sh*:

```
#!/bin/bash

echo "Starting program at $(date)"
# "$#" retrieves the number of arguments
echo "Running program $0 with $# files"
for file in "$@"; do
    echo "- Toren Fronsdal" >> "$file"
done
```

We can then give execute permissions to the script and run it:

```
$ chmod u+x append_signature.sh
$ ./append_signature.sh file1.txt file2.txt file3.txt
```

Further Resources

A Practical Guide to Linux Commands, Editors and Shell Programming by Mark G. Sobell (2013)

The Missing Semester of Your CS Education (MIT Course)

Practical Unix (Stanford Course)