Linux Introduction

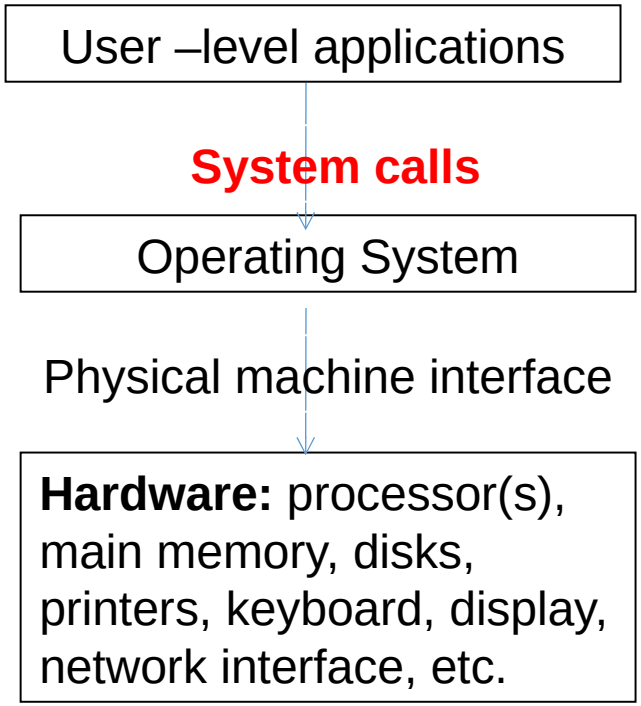
EECS 2031

Gias Uddin
guddin@yorku.ca
https://giasuddin.ca

Acknowledgement

- Some of the covered materials are based on previous EECS2031 offerings:
  - Uyen Trang (UT) Nguyen, Pooja Vashisth, Hui Wang, Manos Papagelis, Song Wang

What is Operating System?



- OS is system software that manages computer hardware and software resources and provides common services for computer programs.

- From app. programmer's point of view:
  - O.S. manages hardware resources
  - O.S. provides user programs with a simpler interface, i.e. system calls
    - cnt=read(fd, buffer,nbytes)
    - getc() etc.

Kernel of Operating System

- Operating system:
  - The entire package consists of central software managing a computer's resources and the accompanying standard software tools, such as command-line interpreters, graphical user interfaces, file utilities, and editors.
  - **kernel**: central software that manages and allocates computer resources (i.e., CPU, RAM, and devices).

Kernel Functionalities: Process scheduling

- Managing one or more central processing units (CPUs)
  - A process is essentially running software.

- Unix: a preemptive multitasking operating system
  - multiple processes (i.e., running programs) can simultaneously reside in memory and each may receive use of the CPU(s).
  - Preemptive: scheduler can preempt (or interrupt) a process, and resume its execution later => to support interactive responses
    - the processors are allowed to spend finite chunks of time (*quanta, or timeslices*) per process

Kernel Functionalities: Memory management

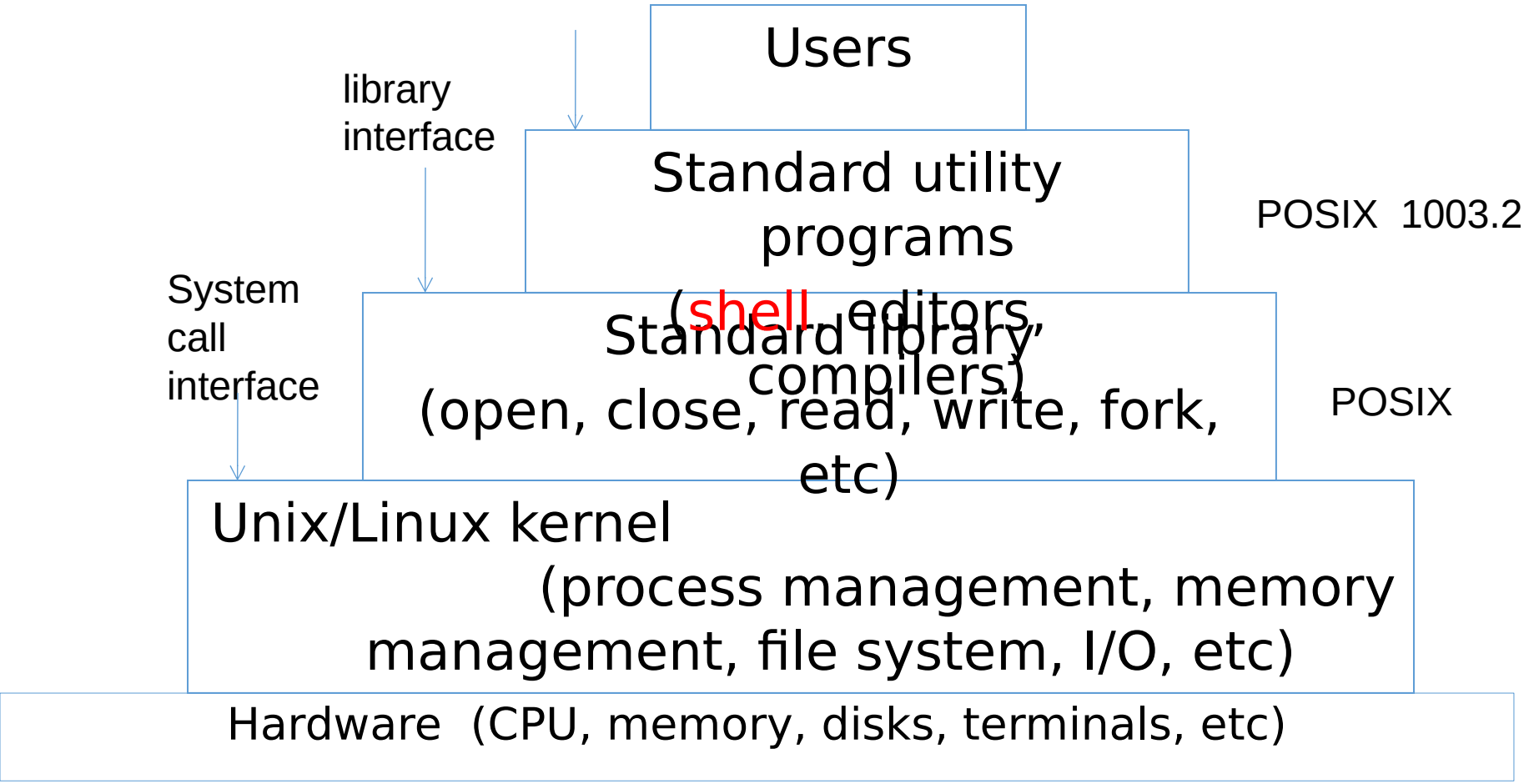
- Manage physical memory (RAM) to be shared among processes in an equitable and efficient fashion

- Virtual memory management:
  - Processes are isolated from one another and from the kernel so that one process can't read or modify the memory of another process or the kernel.
  - Only part of a process needs to be kept in memory, thereby lowering the memory requirements of each process and allowing more processes to be held in RAM simultaneously.
  - better CPU utilization, since it increases the likelihood that, at any moment in time, there is at least one process that the CPU(s) can execute.

Other OS functionalities …

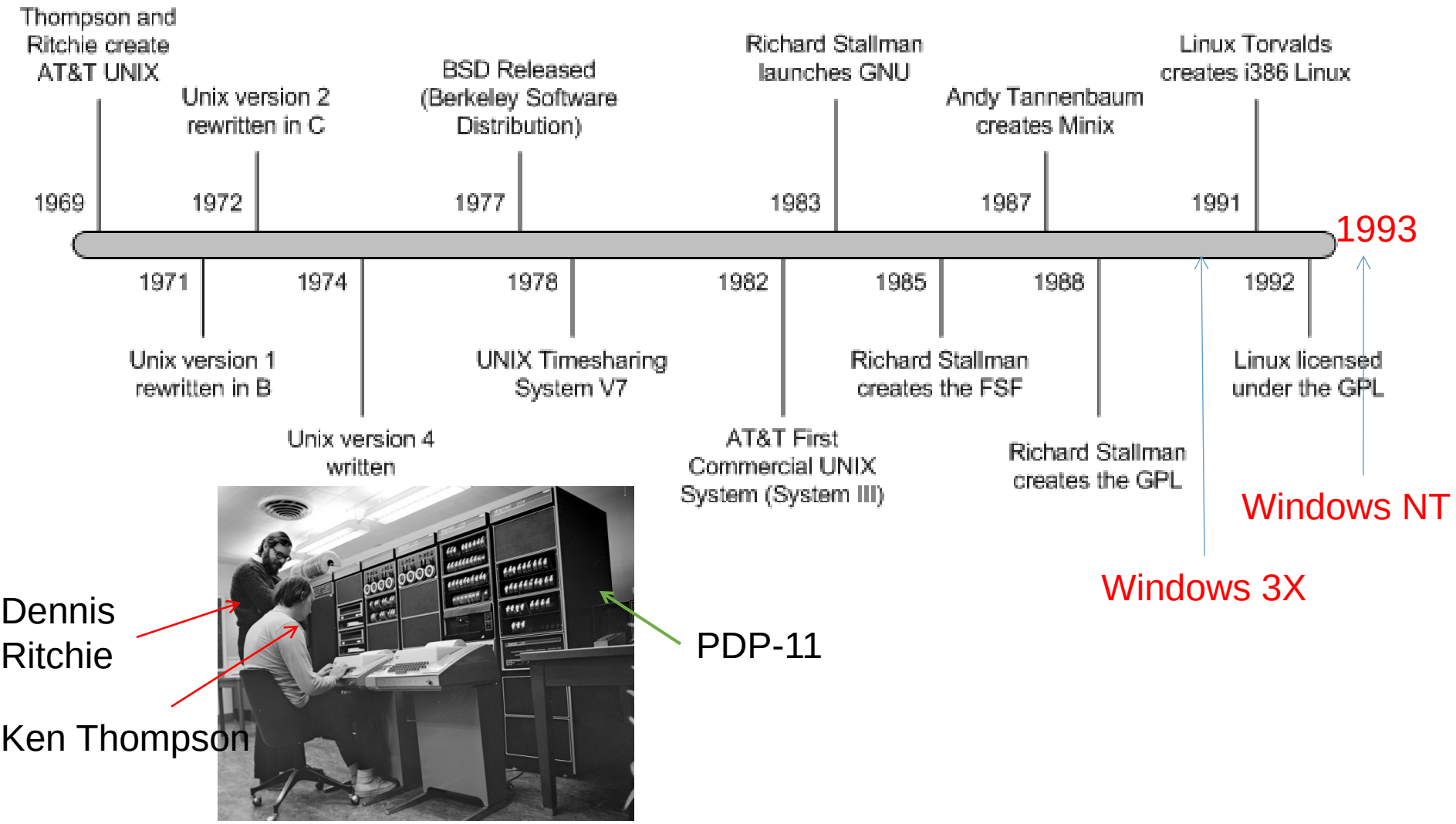
- The kernel provides a file system on disk, allowing files to be created, retrieved, updated, deleted, and so on.
- Creation and termination of processes
- Peripheral device: standardizes and simplifies access to devices, arbitrates access by multiple processes to each device
- Networking: transmits and receives network packets on behalf of user processes.
- Support system call interfaces: processes can request the kernel to perform various tasks using kernel entry points known as system calls.
  - Second part of this course: Unix system call API

Layers in UNIX/Linux System



POSIX, "**P**ortable **O**perating **S**ystem **I**nterface", is a family of standards specified by IEEE for maintaining compatibility between Unix

Timeline of Unix/Linux, GNU



What is Linux?

- Linux is a Unix clone written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net.

- Unix is a multitasking, multi-user computer operating system originally developed in 1969 by a group of AT&T employees at Bell Labs.

- Linux and Unix strive to be POSIX compliant.

- **>60% of the world's servers run some variant of Unix or Linux**. The Android phone and the Kindle run Linux.

Linux Has Many Distributions

What is Linux?
Linux + GNU Utilities = Free Unix

- Linux is an O/S core written by Linus Torvalds and others AND

- a set of small programs written by Richard Stallman and others. They are the GNU utilities.

http://www.gnu.org/

GNU history

- GNU: a free UNIX-like operating system
- **Richard Matthew Stallman** (author of Emacs

other utilities, ls, cat, …, on linux)

  - 1983: development of a free UNIX-like operating system
  - Free Software Foundation (100s of Programmers)

- Free software:
  - freedom to run the program, for any purpose.
  - freedom to study how the program works and adapt it to your needs.
  - freedom to redistribute copies so you can help others.
  - freedom to improve the program and release your improvements to the public, so that everyone benefits.

GPL License

- GNU **General Public License** is a free, copyleft license for software and other kinds of works…
  - "The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the **GNU General Public License is intended to guarantee your freedom to share and change all versions of a program-**-to make sure it remains free software for all its users."
- Manual pages for commands include copyright info:

COPYRIGHT

Copyright © 2011 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.

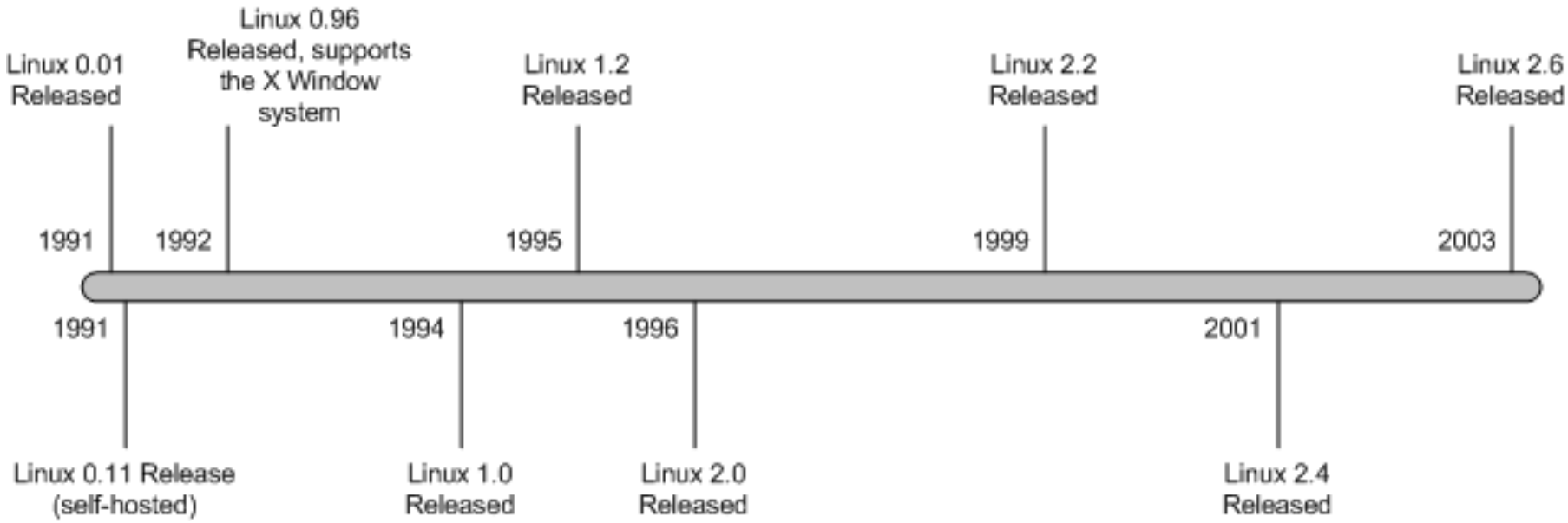
This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

Linux history

- Linus Torvalds
  - 1991: “hobby” operating system for i386-based computer, while study in Univ. of Hel
- 1996: Linux becomes a GNU software component
- **GNU/Linux**: A fairer name than Linux?
  - “Most operating system distributions based on Linux as kernel are basically modified versions of GNU operating system. We began developing GNU in 1984, years before Linus Torvalds started to write his kernel. Our goal was to develop a completely free operating system. Of course, we did not develop all the parts ourselves—but we led the way. We developed most of the central components, forming the largest single contribution to the whole system. The basic vision was ours too. “ --- RMS

https://www.gnu.org/gnu/gnu-linux-faq.en.html

Linux kernel versions



Use “uname –a” to check system information (including kernel version).

Understanding your Linux details

$ **uname -a**

Linux indigo1 4.18.0-513.9.1.el8_9.x86_64 #1 SMP Wed Nov 29 18:55:19 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux

Kernel name: Linux

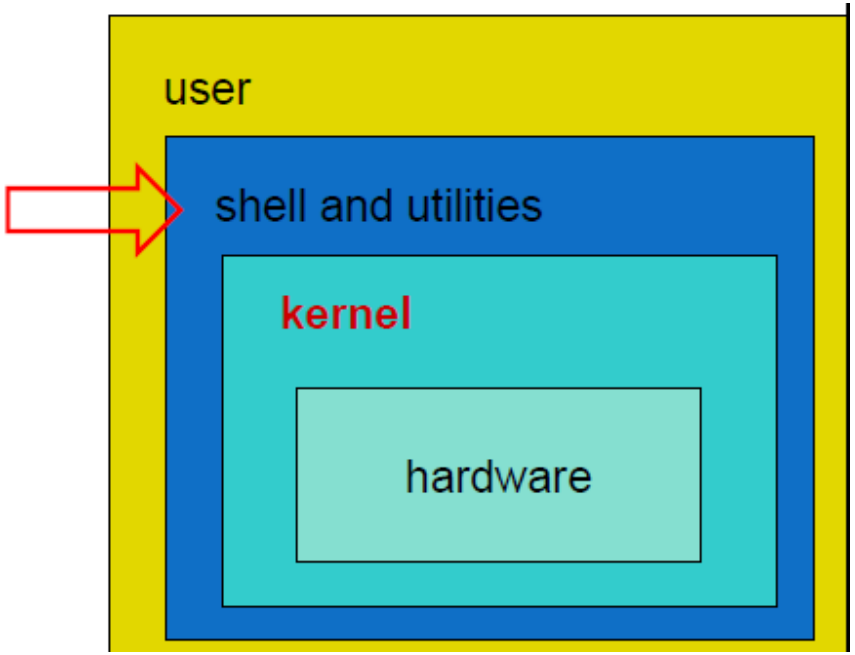
- Hostname: indigo1
- Kernel release: 4.18.0-513.9.1.el8_9.x86_64
- Kernel version: #1 SMP Wed Nov 29 18:55:19 UTC 2023
- Machine hardware name: x86_64
- Processor:x86_64
- Operating system: GNU/Linux

Shell

- Shell: a special-purpose program, **command line interpreter**, reads commands typed by a user and executes programs in response to entered commands

- Many different shells:
  - Bourne Shell (sh): oldest,
    - I/O redirection, pipelines, filename generation (globbing), variables, environment variables, command substitution, background command execution, function
  - C Shell (csh): syntax of flow-control similar to C, command history, command-line editing, job control, aliases
  - Korn Shell (ksh): “csh”, compatible with sh
  - Bourne again Shell (bash): GNU’s reimplementation of Bourne shell, supports features added in C shell, and Korn shell

Shell and Utilities



- Command-line utilities\shell commands e.g., cd ls mkdir ….
- Scripting
  - A set of shell commands that constitute an executable program: a script
  - Batch file .bat in Windows

```
date
ls
…..
```

Check/Change Login Shell

- To check the shell you are using
  - echo $SHELL
  - echo $0

```
indigo 304 % echo $SHELL
/cs/local/bin/tcsh
indigo 305 %
```

- login shell: default shell for a user, specified in /etc/passwd
- To change your login shell, use the command
  - chsh

Check/Change Login Shell

- To check the shell you are using
  - echo $SHELL
  - echo $0

```
indigo 304 % echo $SHELL
/cs/local/bin/tcsh
indigo 305 %
```

- login shell: default shell for a user, specified in /etc/passwd
- To change your login shell, use the command
  - chsh

```
indigo1 60 % chsh
Changing shell for guddin
Available shells:
        /cs/local/bin/tcsh
        /cs/local/bin/bash
        /cs/local/bin/sh
        /cs/local/bin/csh
        /cs/local/bin/ksh
Old shell: /cs/local/bin/tcsh
New shell: /cs/local/bin/bash
Your shell has been changed to /cs/local/bin/bash
indigo1 61 % echo $SHELL
/cs/local/bin/tcsh
indigo1 62 % echo $0
/cs/local/bin/tcsh
indigo1 63 %
```

Shell: interactive mode

- A shell session (a dialog between user and shell)
  1. Displays a **prompt** character, and waits for user to type in a command line
     - Prompt depends on shell: sh, ksh, bash: $ csh: % tcsh: >
     - May be customized (with current directory, host, …)
  2. On input of a command line, shell extracts **command name and arguments**, searches for the program, and runs it.
  3. When program finishes, shell continues to step 1
  4. The loop continues until user types “exit” or “ctrl-d” to end

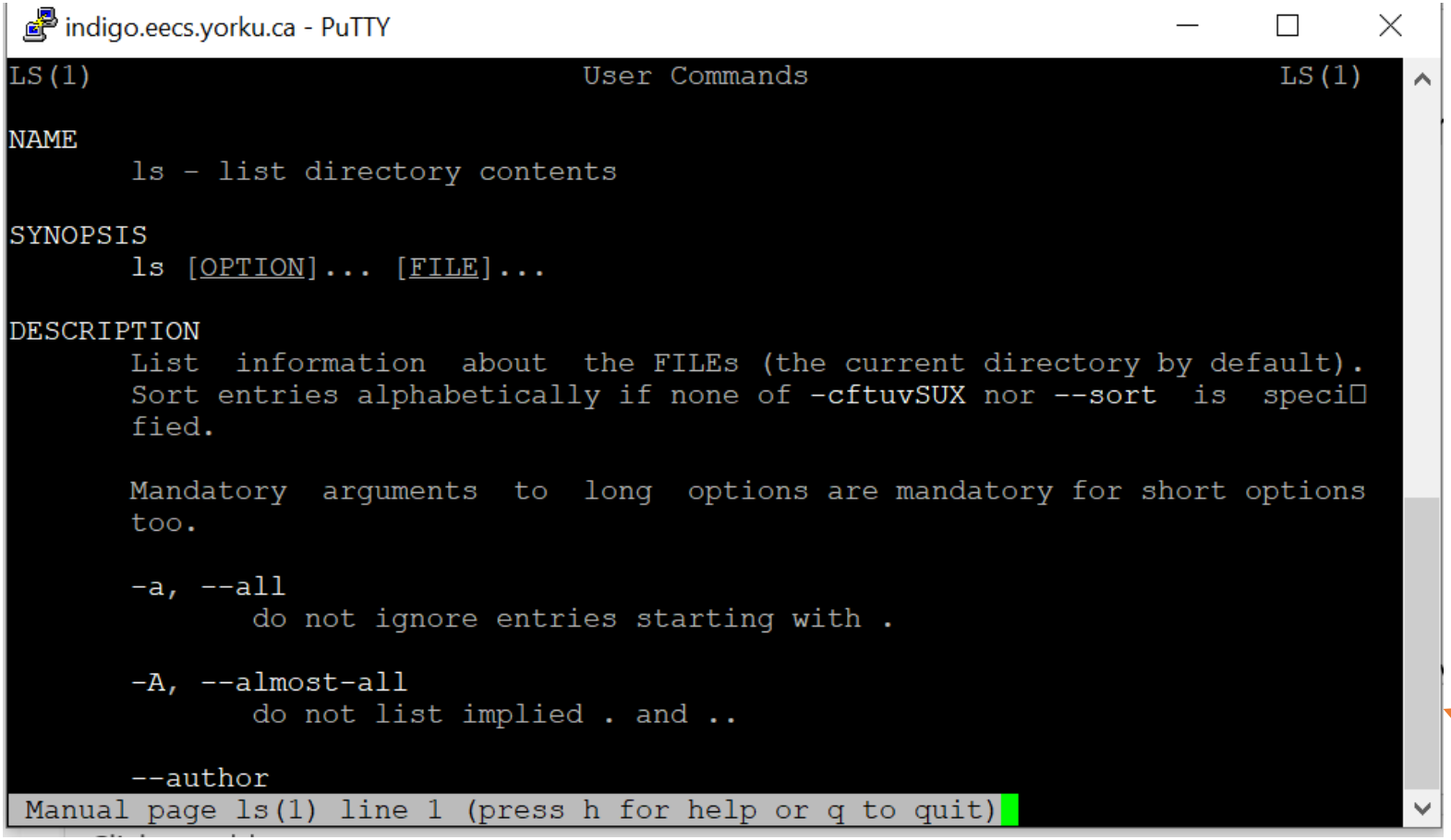
UNIX command line

- Command name and arguments:

command [[-] option (s)] [option argument (s)] [command argument (s)]

  - Command arguments are mostly file or directory names
    - cp prog1.c prog1.c.bak
  - Options: used to control the behavior of the command
    - *head -20 lab1A.c*
    - *wc –w lab2.c* // count how many words
    - Some options come with option argument
      - *sort –k 1 data.txt*
      - // use the first column of data.txt as the key to sort

The most important command !!!



man ls

- man: displaying online manuals
  - Press q to quit, space to scroll down, arrow keys to roll up/down

Correcting type mistakes

- Shell starts to parse command line only when **Enter** key is pressed
- Delete the whole line (line-kill): **Ctrl-u**
- Erase a character: **Ctrl-h** or backspace key
- Many more fancy functionalities:
  - Auto-completion: **press Tab key to ask shell to auto-complete command, or path name**
  - History (repeat command): use arrow (up and down) keys to navigate past commands
  - …

Shell: batch/scripting mode

- In batch mode, shell can interpret and execute shell scripts

```
#!/bin/bash
# count number of files/directories in curr. directory
ls –l | wc –l
```

- Shell constructs:
  - variables,
  - Loop and conditional statements
  - I/O commands (read from keyboard, write to terminal)
  - Function, arrays …

Unix File

- Files: store information
  - a sequence of **0 or more bytes** containing arbitrary information

- What's in a filename?
  - Case matters; the limitation is 255 bytes

  - Special characters such as -, and spaces are allowed, but you shouldn’t use them in a filename
    - Can you think of the reasons ?

  - Dot files are hidden, i.e., normally not listed by command ***ls***
    - To display all files, including hidden files, use ls -a

What's in a file?

- So far, we learnt that files are organized in a hierarchical directory structure
  - Each file has a name, resides under a directory, is associated with some admin info (**permission, owner**)
- Contents of file:
  - Text (ASCII) file (such as your C/C++ source code)
  - Executable file (commands)
  - A link to other files, …
  - Virtual file:
    - /proc: a pseudo-filesystem, contains user-accessible objects on runtime state of kernel and executing processes
- To check the type of file: "*file <filename>*"

```
indigo 312 % ls
1          countChar
1.t        countChar.c
```

```
indigo 313 % file countChar.c
countChar.c: C source, ASCII text
```

```
indigo 314 % file countChar
countChar: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically link
ed, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/Linux 3.2.0, BuildID[sha1]=
a0a304a94e2c8846f2de5b7050c64c642b766abf, not stripped
```

File Viewing Commands

- **cat**: concatenate files and display on standard output (i.e., the terminal window)
  - *cat [option] … [file] …*
  - *cat proj1.c*
  - *cat proj1.c proj2.c*
  - *cat –n proj1.c // display the file with line #*

[ ] means the argument is optional
… means there can be multiple arguments of this type

- **more**: file perusal filter (i.e., displaying file one screen at a time)
  - *more proj1.cpp*

- **head, tail**: display the beginning or ending lines of a file
  - *tail -f output // display the file, append more lines as the file grows*

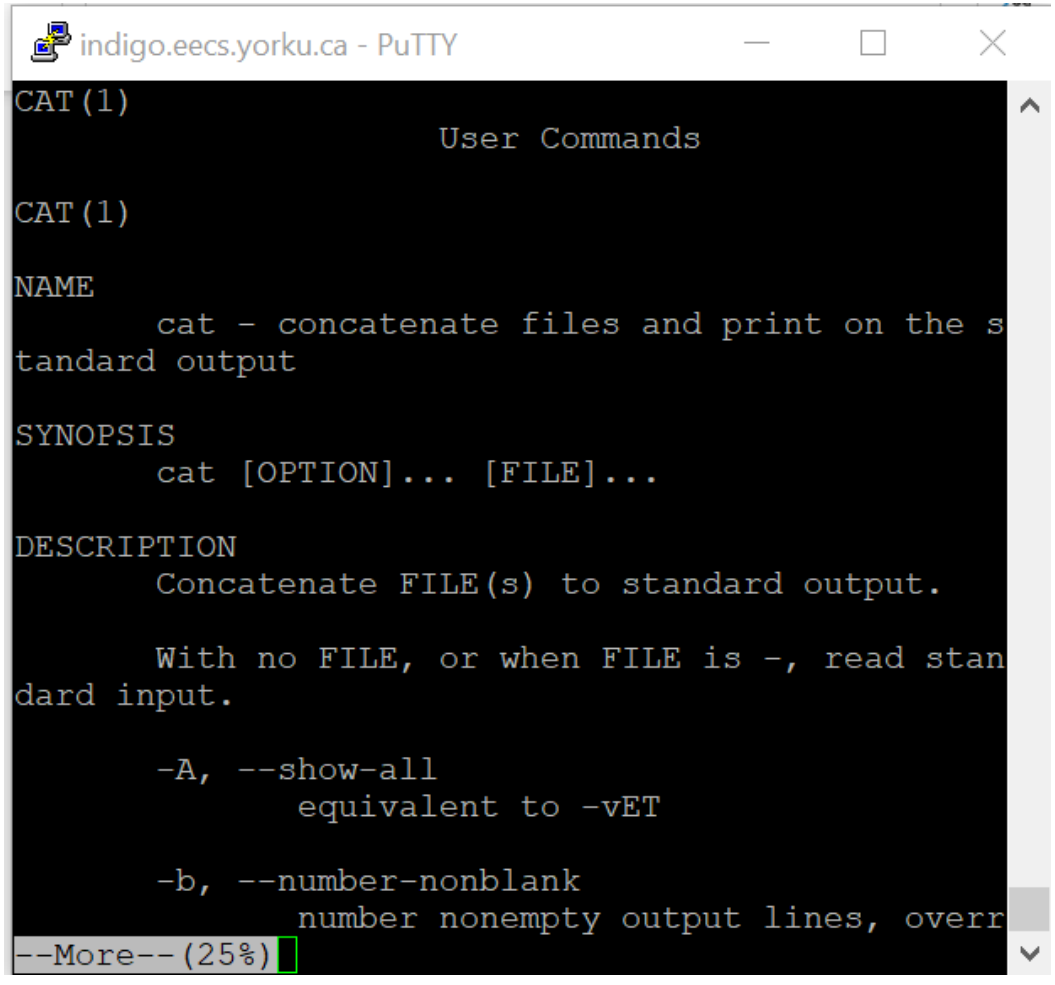
```
CAT(1)                          User Commands                          CAT(1)

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION]... [FILE]...

DESCRIPTION
       Concatenate FILE(s) to standard output.

       With no FILE, or when FILE is -, read standard input.

       -A, --show-all
              equivalent to -vET

       -b, --number-nonblank
              number nonempty output lines, overrides -n

       -e     equivalent to -vE

       -E, --show-ends
              display $ at end of each line

       -n, --number
              number all output lines

       -s, --squeeze-blank
              suppress repeated empty output lines

       -t     equivalent to -vT

       -T, --show-tabs
              display TAB characters as ^I

       -u     (ignored)

       -v, --show-nonprinting
              use ^ and M- notation, except for LFD and TAB

       --help display this help and exit

       --version
```

```
indigo 316 % cat countChar.c
#include<stdio.h>

int main(){
    int c;
    int count = 0;

    c= getchar();
    while(c !=EOF){
        count++;
        c =getchar();
    }
    printf("# of chars: %d\n", count);
}
```

```
indigo 318 % cat -n countChar.c
     1  #include<stdio.h>
     2
     3  int main(){
     4      int c;
     5      int count = 0;
     6
     7      c= getchar();
     8      while(c !=EOF){
     9          count++;
    10          c =getchar();
    11      }
    12      printf("# of chars: %d\n", count);
    13  }
    14
```

```
indigo 323 % man cat > cat.man
```

```
indigo 325 % more cat.man
```



```
indigo 326 % head cat.man
CAT(1)                    User Commands

CAT(1)

NAME
       cat - concatenate files and print on the
tandard output

SYNOPSIS
       cat [OPTION]... [FILE]...

DESCRIPTION
       Concatenate FILE(s) to standard output.
indigo 327 %
```

- First n (on default 10) line

```
indigo 330 % head -2  cat.man
CAT(1)                                                    User Commands

indigo 331 %
```

```
indigo 328 % tail cat.man
       Copyright © 2018 Free Software Foundation, Inc.  Licen
       This is free software: you are free to change and redi

SEE ALSO
       tac(1)

       Full documentation at: <https://www.gnu.org/software/c
       or available locally via: info '(coreutils) cat invoca

GNU coreutils 8.30
```

- Last n (on default 10) line

```
indigo 329 % tail -2  cat.man

GNU coreutils 8.30                                   July 201
```

File manipulation commands

- **rm**: remove one or multiple files or directories
  - *rm [option] … FILE …*
  - *rm temp*
  - *rm temp1 temp2*

- Wildcards (metacharacter) can be used in the command line
  - Letter * matches with any string
    - *rm *.o: remove all .o files* **(be careful !!!)**
  - ?: match any one character
  - [abc]: match with letter a or b or c

- *rm –r*: remove directories and their sub-dirs recursively
- *rm –i* : confirm with user before removing files

File manipulation commands (2)

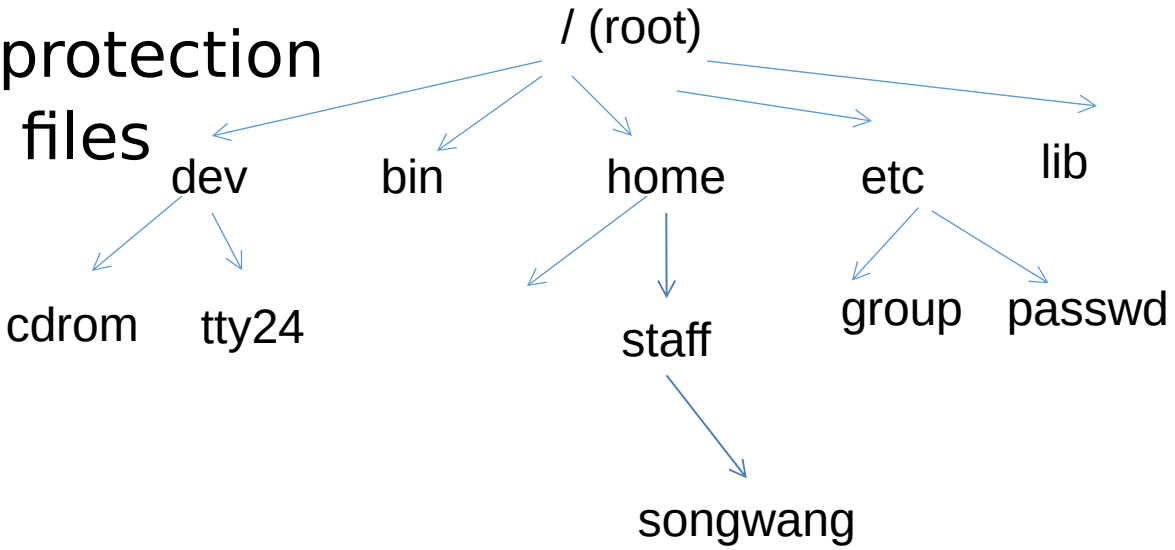
- cp: copy file or directory
  - *cp [OPTION] SOURCE DESTINATION*
- To make a backup copy of your program before dramatic change
  - ***cp proj1.c proj1.c.bak***
- To make a backup copy of a whole directory
  - *cp **–r** lab1_dir lab1_dir_backup*
  - *-R, -r, --recursive*: copy directories recursively

File manipulation commands (3)

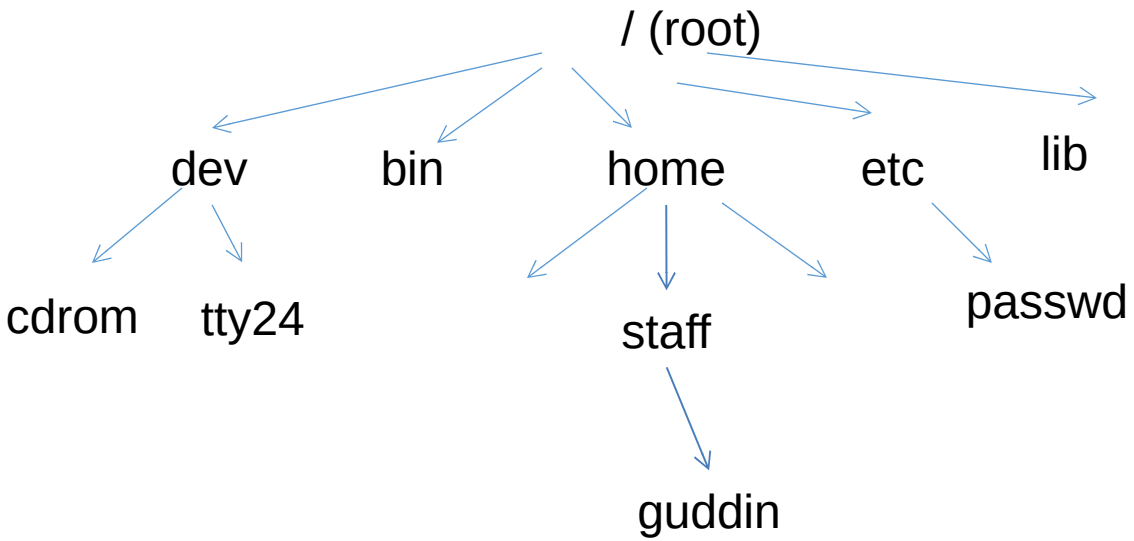
- mv: move (rename) files/directories
  - *mv [OPTION] SOURCE DEST*
    - Rename SOURCE to DEST
    - *mv proj1.c lab1.c*
  - *mv [OPTION]… SOURCE… DIRECTORY*
    - Move SOURCE to DIRECTORY
    - *mv lab1.c lab2.c EECS2031*

Hierarchical file system

- Directory: a file that can hold other files
- Advantages of hierarchical file system:
  - Files can have same names, as long as they are under different directories
  - Easier for protection
  - Organized files



Absolute pathname, path



- Pathname of a file/directory: location of file/directory in the file system
  - How do you tell other where your prog. Is located ?
- Absolute pathname: path name specified relative to root, i.e., starting with the root (/)
  - e.g., /home/staff/songwang
  - What's the absolute pathname for the "passwd" file?

Home directory

- Every user has a home directory created for him/her
  - When you log in, you are in your home directory
  - In home directory, a user usually has permission to create files/directories, remove files ..
  - ~ to refer to current user's home directory
  - ~username to refer to username's home directory

Current directory & Relative Pathname

- Tiring to specify absolute pathname each time
- To make life easier: working directory
  - User can move around the file system, shell remembers where the user is (i.e., current directory)
- To check your current directory, use command:

```
indigo1 48 % pwd
/eecs/home/guddin/EECS2031
indigo1 49 % ~
```

Getting around in the file system

- To create a subdirectory:
  - *mkdir [option]… directory…*
  - cd
  - mkdir labtest2
  - cd labtest2
  - mkdir question

- To remove a directory:
  - *rmdir [option]… directory…*
  - Report failure if directory is not empty
    - Can use rm –rf to remove non-empty directory

Relative pathname

- Absolute pathname: specified relative to root
- Relative pathname: specified relative to current directory
  - . (current directory), .. (parent directory, one level up)
  - If current directory is at /home/staff/zhang, what is the relative pathname of the file passwd?
    - *../../../etc/passwd*: go one level up, go one level up, go one level up, go to etc, passwd is there