ICCS207: Term I/2018-19

Introduction to Computer Systems

~~Introduction to File Processing~~

Sunsern Cheamanunkul (sunsern.che@mahidol.edu)

Kritya Bunchongchit (kritya.bun@mahidol.edu)

Course Overview

- Instructors:
  - Sunsern Cheamanunkul (sunsern.che@mahidol.edu)
  - Kritya Bunchongchit (kritya.bun@mahidol.edu)
- Office Hours: TBA on Canvas
- Required Textbooks: [Not really]
- Supplemental Reading: TBA on Canvas

Course Website

https://canvas.instructure.com/enroll/JCRXCM

- Sign up for Canvas and enroll the course by using the above URL.
- We will use this for announcements, homework submission, grade book, etc.

Why System-level?

- Most CS and CE courses emphasize abstraction
  - Abstract data types
  - Asymptotic analysis
- These abstractions have limits
  - Especially in the presence of bugs
  - Need to understand details of underlying implementations
- Useful outcomes
  - Become more effective programmers
    - Able to find and eliminate bugs efficiently
    - Able to understand and tune for program performance
  - Prepare for and recap “systems” classes:
    - Operating Systems, Computer Architecture, Backend Tech

Course Outline

- Module I: Working in Linux environment
- Module II: C Programming
- Module III: Representation
- Module IV: Memory organization and management

Module I: Working in Linux Environment

- Practical skills you will find them useful for the rest of your life…
- Topics include:
  - Basic Linux shell commands
  - Shell scripting
  - String processing tools
  - Git
  - etc.

Module II: C Programming

- Learn to code in C
    - and really understand how C pointers work.
- Learn to use build/debug tools
- We will not cover C++

Module III: Representation

- You will learn about how computers store data and how they operate.
- Topics include:
  - Data representation
  - Assembly language

Module IV: Memory Organization and Management

- You will learn about how computer manages memory and what really happens when a program runs
- Topics include:
  - Caching
  - Virtual Memory
  - Dynamic memory allocation

Tentative Schedule

Week 1-3: Module I: Working in Linux

— Quiz 1 —

Week 4-7: Module II: C programming

— Quiz 2 —

Week 9-10: Module III: Representation

— Quiz 3 —

Week 11-12: Module IV: Memory organization

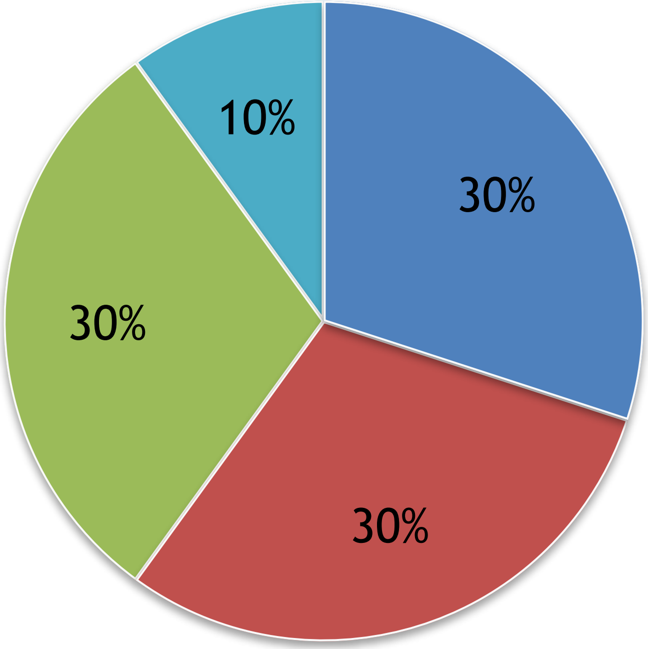
— Final Exam —

Course Structure

- 24 lectures
    - Bring your laptops for in-class activities
    - Attendance is **strongly recommended**
- ~4 assignments
    - Programming/computer-based assignments
    - 2 late tokens for the course. Max of 1 token can be used per assignment
    - Late homework without token will not be graded

Grading

- 30% - Assignments
- 45% - Quizzes
- 20% - Final
- 5% - Participation



- Per OAA: **A is 90 or more; F is below 60**

Exams

- We don’t have midterm exam. Instead, we have 3 quizzes + 1 Final Exam (which is just another quiz)
- Each quiz will be administered after each module
- Most quizzes will have two parts:
  1. paper-based — you will be tested on the concepts and materials presented in class.
  2. computer-based — you will write programs to solve problems.

Expectations

- You are expected to
  - Take responsibility for the material, homework, exams etc.
  - Work (really) hard.
  - Ask questions to help you learn.
  - Read the assigned reading if any.
  - Engage in the in-class activities

Policy on Collaboration

- Working together is important.
  - Discuss course material in general terms
  - OK to suggest how to debug
- No collaboration whatsoever on quizzes and final.

Cheating

- What is cheating?
  - Sharing code: by copying, retyping, looking at, or supplying a file
  - Coaching your friend to write a lab, line by line
  - Copying code from previous course or from elsewhere on WWW
    - Only allowed to use code we supply
- What is NOT cheating?
  - Explaining how to use systems or tools
  - Helping others with high-level design issues
- Detection of cheating:
  - We can check, really.
  - Our tools for doing this are much better than most cheaters think!

Conflicts

- Conflict exams, other irreducible conflicts
  - OK, but must make PRIOR arrangements with us
  - Notifying us well ahead of time shows maturity and makes us like you more (and thus to work harder to help you out of *your* problem)

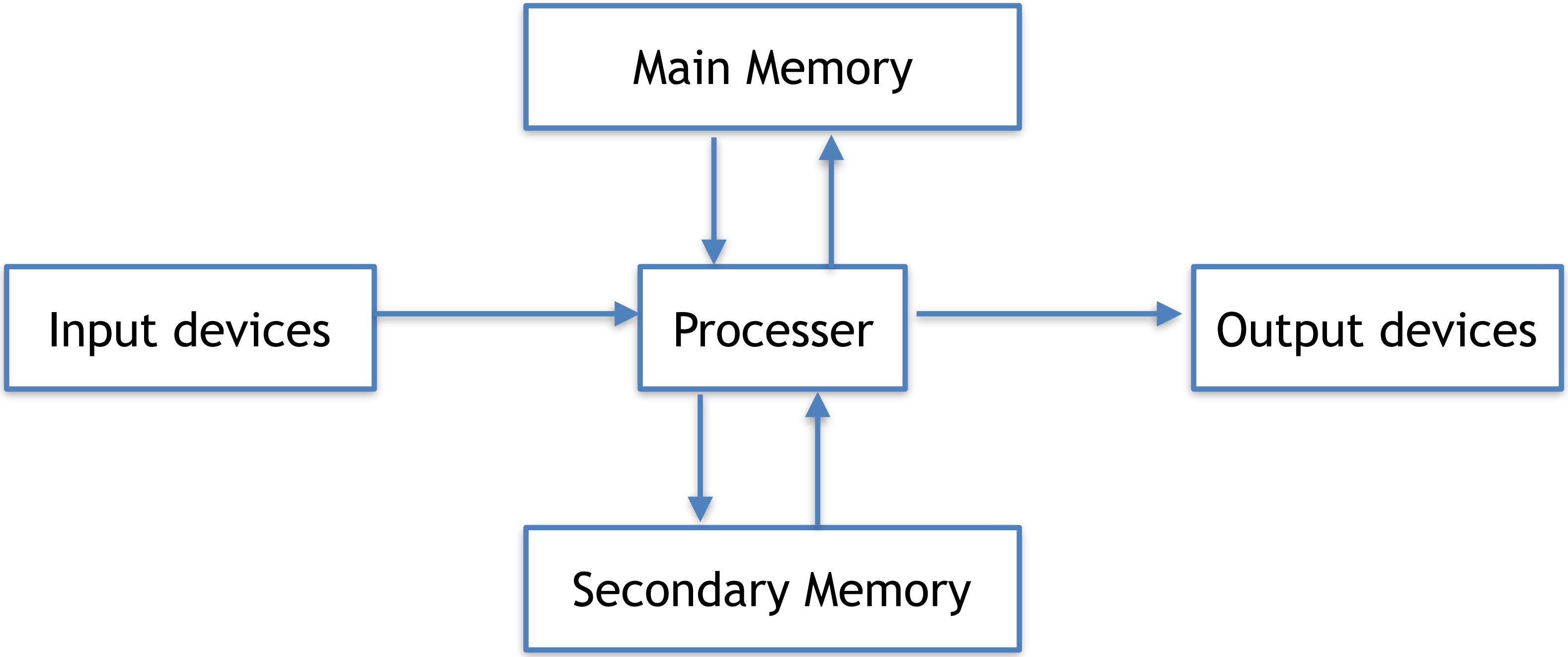
Facilities

- You will be working on our **Hamachi** server.
- Ideally we want you to work remotely on the machine.
  - Don’t worry we will teach do that.
- Currently, Hamachi can be accessed from LAN connections (TTT Wifi) only.

Let’s our journey begin…

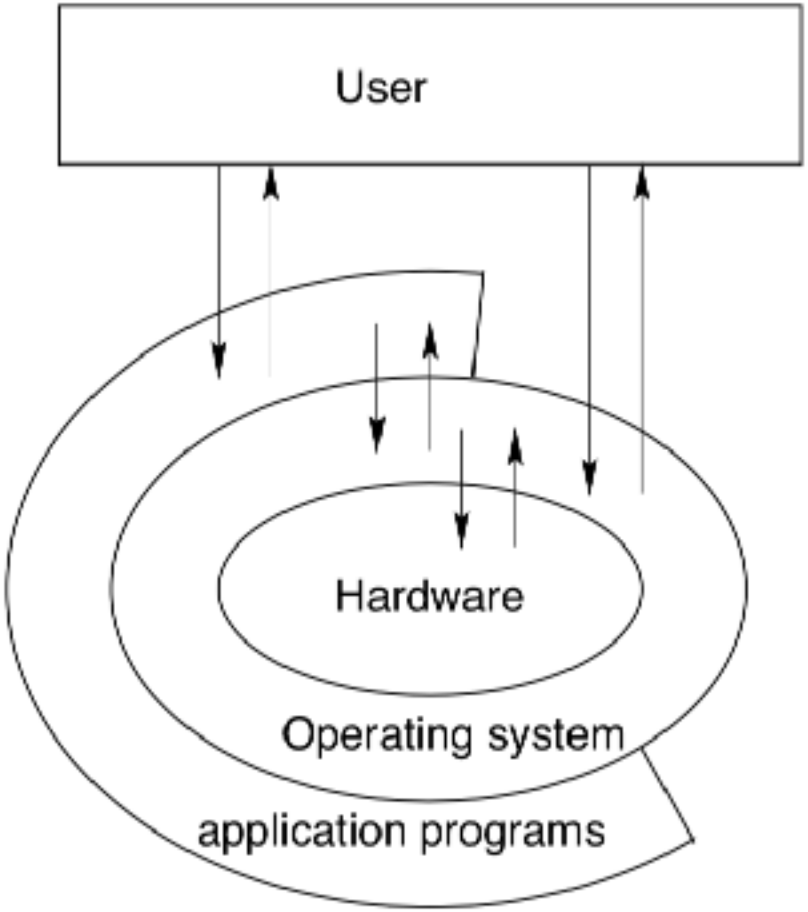
DEC VT100 — It was introduced in 1978

Computer 101



Operating systems

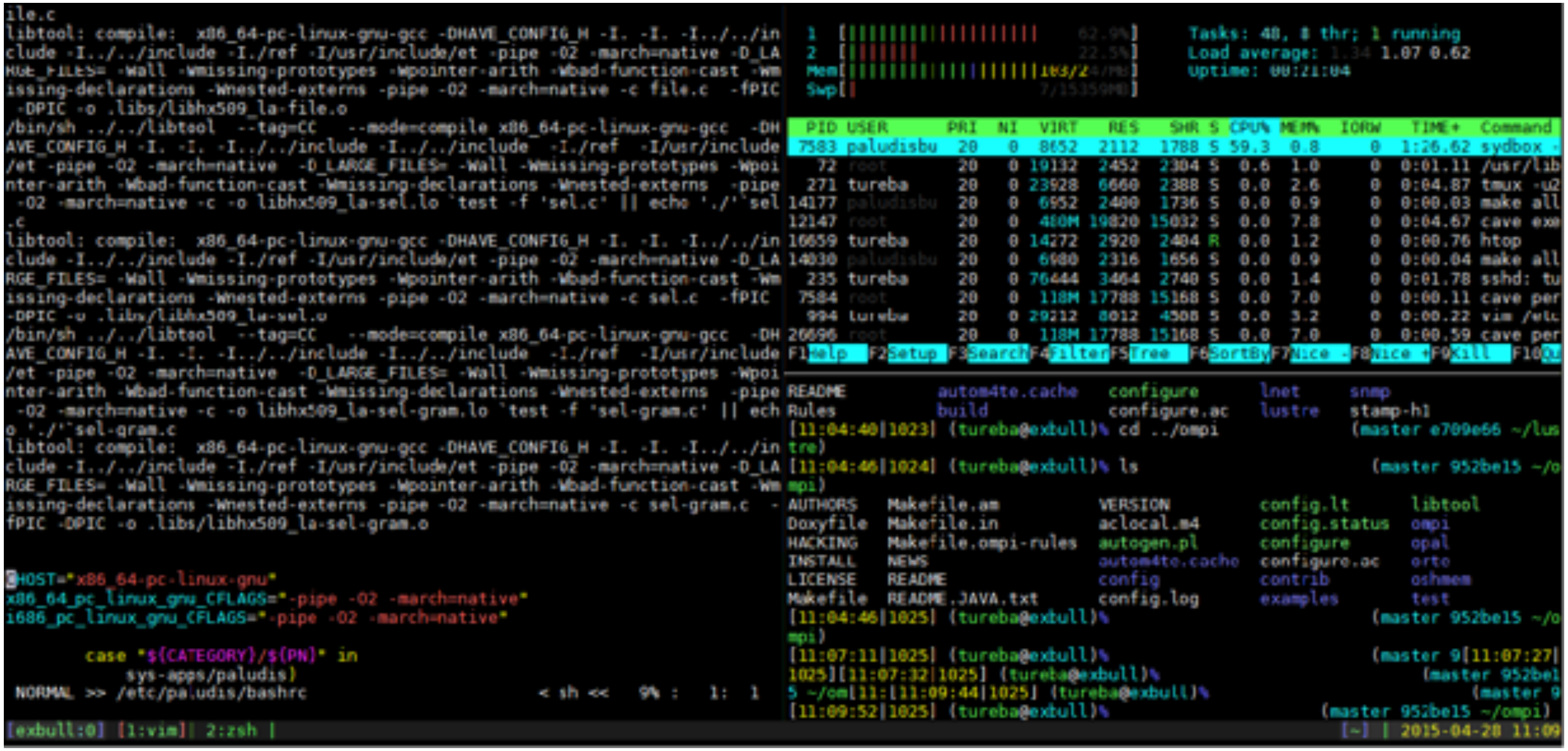
Operating systems



UNIX Operating Systems

- UNIX is a Multi-User/Multi-Tasking operating system and exists in many different versions ("derivates"): Solaris, AIX, XENIX, HP-UX, SINIX, Linux.
- It is mainly used for scientific-technical applications on mainframes and workstations, but has become, because of **Linux,** also popular for classical PC-applications throughout the last years.
- Linux is available in many different distributions e.g. Centos, Ubuntu, Fedora, Debian, Redhat

Why not GUI?

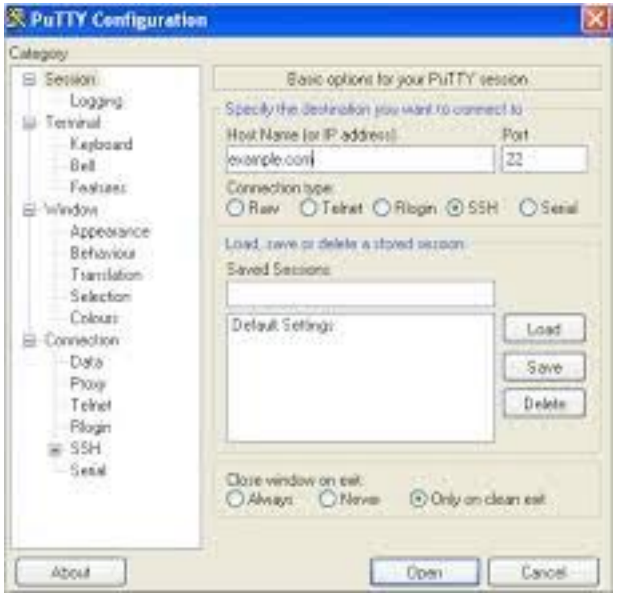


```
ile.c
libtool: compile:  x86_64-pc-linux-gnu-gcc -DHAVE_CONFIG_H -I. -I. -I../../in
clude -I../../include -I./ref -I/usr/include/et -pipe -O2 -march=native -D_LA
RGE_FILES= -Wall -Wmissing-prototypes -Wpointer-arith -Wbad-function-cast -Wm
issing-declarations -Wnested-externs -pipe -O2 -march=native -c file.c  -fPIC
 -DPIC -o .libs/libhx509_la-file.o
/bin/sh ../../libtool  --tag=CC   --mode=compile x86_64-pc-linux-gnu-gcc  -DH
AVE_CONFIG_H -I. -I. -I../../include -I../../include  -I./ref  -I/usr/include
/et -pipe -O2 -march=native  -D_LARGE_FILES= -Wall -Wmissing-prototypes -Wpoi
nter-arith -Wbad-function-cast -Wmissing-declarations -Wnested-externs  -pipe
 -O2 -march=native -c -o libhx509_la-sel.lo `test -f 'sel.c' || echo './'`sel
.c
libtool: compile:  x86_64-pc-linux-gnu-gcc -DHAVE_CONFIG_H -I. -I. -I../../in
clude -I../../include -I./ref -I/usr/include/et -pipe -O2 -march=native -D_LA
RGE_FILES= -Wall -Wmissing-prototypes -Wpointer-arith -Wbad-function-cast -Wm
issing-declarations -Wnested-externs -pipe -O2 -march=native -c sel.c  -fPIC
-DPIC -o .libs/libhx509_la-sel.o
/bin/sh ../../libtool  --tag=CC   --mode=compile x86_64-pc-linux-gnu-gcc  -DH
AVE_CONFIG_H -I. -I. -I../../include -I../../include  -I./ref  -I/usr/include
/et -pipe -O2 -march=native  -D_LARGE_FILES= -Wall -Wmissing-prototypes -Wpoi
nter-arith -Wbad-function-cast -Wmissing-declarations -Wnested-externs  -pipe
 -O2 -march=native -c -o libhx509_la-sel-gram.lo `test -f 'sel-gram.c' || ech
o './'`sel-gram.c
libtool: compile:  x86_64-pc-linux-gnu-gcc -DHAVE_CONFIG_H -I. -I. -I../../in
clude -I../../include -I./ref -I/usr/include/et -pipe -O2 -march=native -D_LA
RGE_FILES= -Wall -Wmissing-prototypes -Wpointer-arith -Wbad-function-cast -Wm
issing-declarations -Wnested-externs -pipe -O2 -march=native -c sel-gram.c  -
fPIC -DPIC -o .libs/libhx509_la-sel-gram.o


CHOST="x86_64-pc-linux-gnu"
x86_64_pc_linux_gnu_CFLAGS="-pipe -O2 -march=native"
i686_pc_linux_gnu_CFLAGS="-pipe -O2 -march=native"

        case "${CATEGORY}/${PN}" in
            sys-apps/paludis)
 NORMAL >> /etc/paludis/bashrc                             < sh <<   9% :   1:  1
[exbull:0] [1:vim]| 2:zsh |
```

```
  1  [||||||||||||||||||||      62.9%]     Tasks: 48, 8 thr; 1 running
  2  [|||||||                   22.5%]     Load average: 1.34 1.07 0.62
  Mem[|||||||||||||||||||||103/244MB]      Uptime: 00:21:04
  Swp[|                     7/15359MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%  IORW   TIME+  Command
 7583 paludisbu  20   0  8652  2112  1788 S 59.3  0.8     0  1:26.62 sydbox -
   72 root       20   0 19132  2452  2304 S  0.6  1.0     0  0:01.11 /usr/lib
  271 tureba     20   0 23928  6660  2388 S  0.0  2.6     0  0:04.87 tmux -u2
14177 paludisbu  20   0  6952  2400  1736 S  0.0  0.9     0  0:00.03 make all
12147 root       20   0  480M 19820 15032 S  0.0  7.8     0  0:04.67 cave exe
16659 tureba     20   0 14272  2920  2404 R  0.0  1.2     0  0:00.76 htop
14030 paludisbu  20   0  6980  2316  1656 S  0.0  0.9     0  0:00.04 make all
  235 tureba     20   0 76444  3464  2740 S  0.0  1.4     0  0:01.78 sshd: tu
 7584 root       20   0  118M 17788 15168 S  0.0  7.0     0  0:00.11 cave per
  994 tureba     20   0 29212  8012  4508 S  0.0  3.2     0  0:00.22 vim /etc
20696 root       20   0  118M 17788 15168 S  0.0  7.0     0  0:00.59 cave per
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Qu
README          autom4te.cache   configure        lnet       snmp
Rules           build            configure.ac     lustre     stamp-h1
[11:04:40|1023] (tureba@exbull)% cd ../ompi                 (master e709e66 ~/lus
tre)
[11:04:46|1024] (tureba@exbull)% ls                           (master 952be15 ~/o
mpi)
AUTHORS   Makefile.am          VERSION          config.lt       libtool
Doxyfile  Makefile.in          aclocal.m4       config.status   ompi
HACKING   Makefile.ompi-rules  autogen.pl       configure       opal
INSTALL   NEWS                 autom4te.cache   configure.ac    orte
LICENSE   README               config           contrib         oshmem
Makefile  README.JAVA.txt      config.log       examples        test
[11:04:46|1025] (tureba@exbull)%                              (master 952be15 ~/o
mpi)
[11:07:11|1025] (tureba@exbull)%                              (master 9[11:07:27|
1025][11:07:32|1025] (tureba@exbull)%                            (master 952be1
5 ~/om[11:[11:09:44|1025] (tureba@exbull)%                           (master 9
[11:09:52|1025] (tureba@exbull)%                        (master 952be15 ~/ompi)
                                                     [~] | 2015-04-28 11:09
```

Terminal

- A **terminal emulator**, **terminal application** is a program that emulates a video terminal within some other display architecture.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): example.com

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

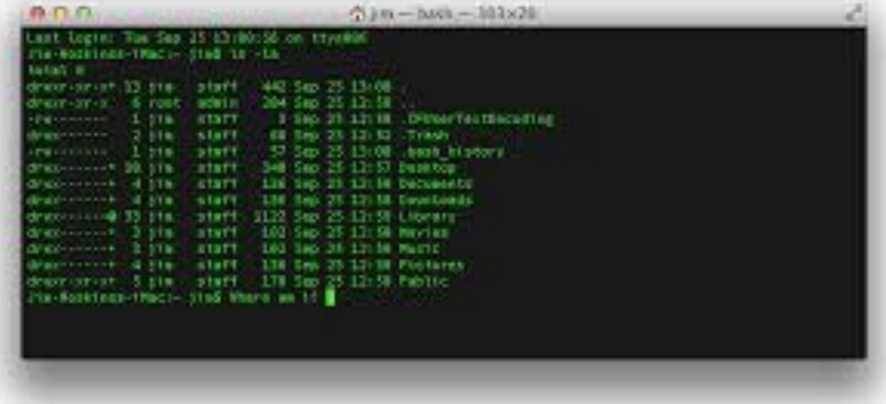
Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

PuTTY for Windows



```
Last login: Tue Sep 25 13:00:56 on ttys000
jim-Bookless-iMac:~ jim$ ls -la
total 0
drwxr-xr-x+ 13 jim   staff   442 Sep 25 13:08 .
drwxr-xr-x   6 root  admin   204 Sep 25 12:58 ..
-rw-------   1 jim   staff     3 Sep 25 12:58 .CFUserTextEncoding
drwx------   2 jim   staff    68 Sep 25 12:52 .Trash
-rw-------   1 jim   staff    57 Sep 25 13:08 .bash_history
drwx------+ 10 jim   staff   340 Sep 25 12:57 Desktop
drwx------+  4 jim   staff   136 Sep 25 12:58 Documents
drwx------+  4 jim   staff   136 Sep 25 12:58 Downloads
drwx------@ 35 jim   staff  1122 Sep 25 12:58 Library
drwx------+  3 jim   staff   102 Sep 25 12:58 Movies
drwx------+  3 jim   staff   102 Sep 25 12:58 Music
drwx------+  4 jim   staff   136 Sep 25 12:58 Pictures
drwxr-xr-x+  5 jim   staff   170 Sep 25 12:58 Public
jim-Bookless-iMac:~ jim$ Where am I?
```

Terminal OSX

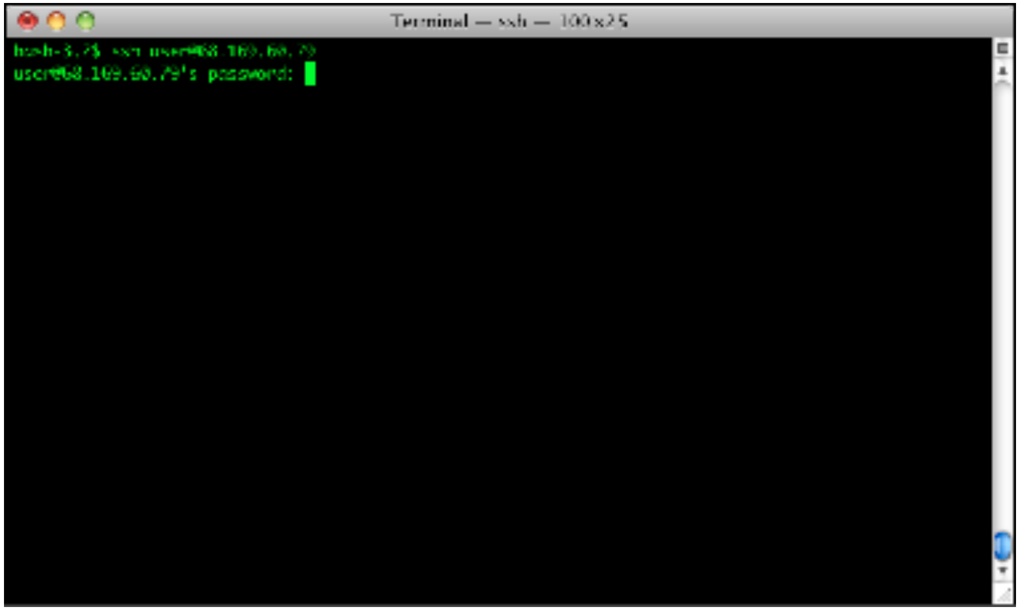
What you see in the movie

Please login

Username: Utopioneer
Error Code: X X X 0 0

Access Denied

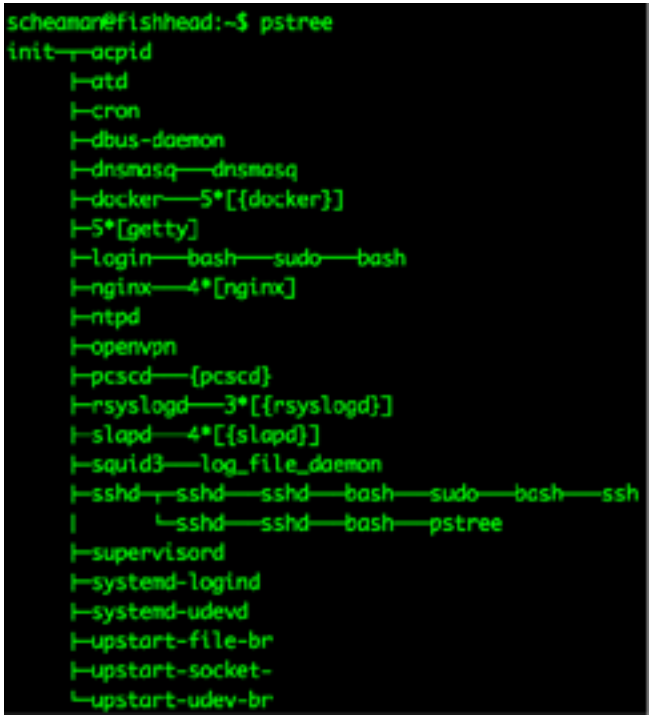
This is reality…



```
bash-3.2$ ssh user@68.109.60.79
user@68.109.60.79's password: 
```

Program vs Process

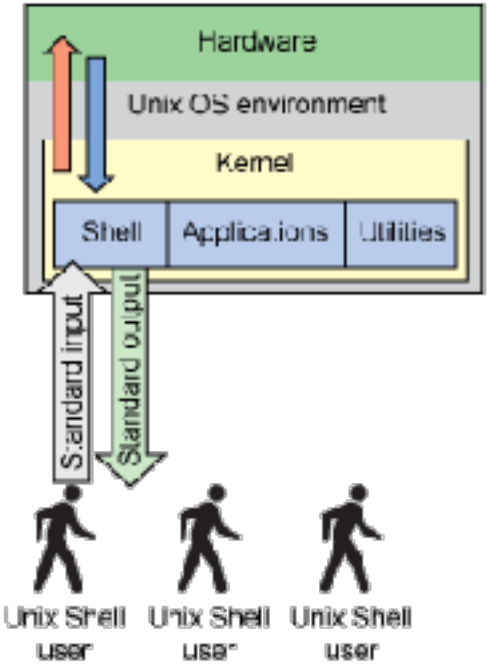
- A program is a sequence of binary data that encodes machine instructions.
- A process is an running instance of a program.
- You can open up multiple terminals. Each terminal runs a shell in a separate process.
- Processes on a machine have a tree structure.



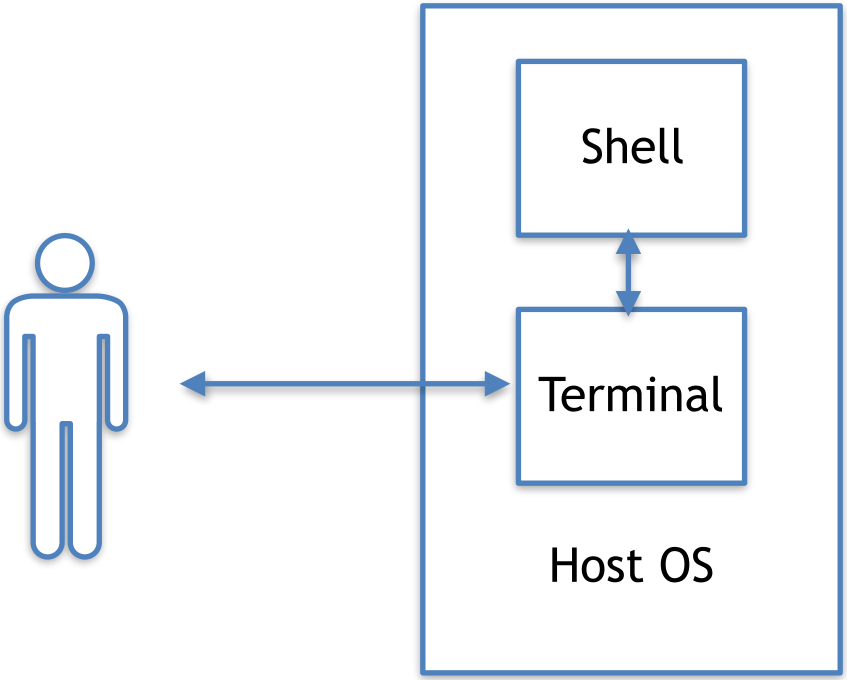
```
scheaman@fishhead:~$ pstree
init─┬─acpid
     ├─atd
     ├─cron
     ├─dbus-daemon
     ├─dnsmasq───dnsmasq
     ├─docker───5*[{docker}]
     ├─5*[getty]
     ├─login───bash───sudo───bash
     ├─nginx───4*[nginx]
     ├─ntpd
     ├─openvpn
     ├─pcscd───{pcscd}
     ├─rsyslogd───3*[{rsyslogd}]
     ├─slapd───4*[{slapd}]
     ├─squid3───log_file_daemon
     ├─sshd─┬─sshd───sshd───bash───sudo───bash───ssh
     │      └─sshd───sshd───bash───pstree
     ├─supervisord
     ├─systemd-logind
     ├─systemd-udevd
     ├─upstart-file-br
     ├─upstart-socket-
     └─upstart-udev-br
```

Shell

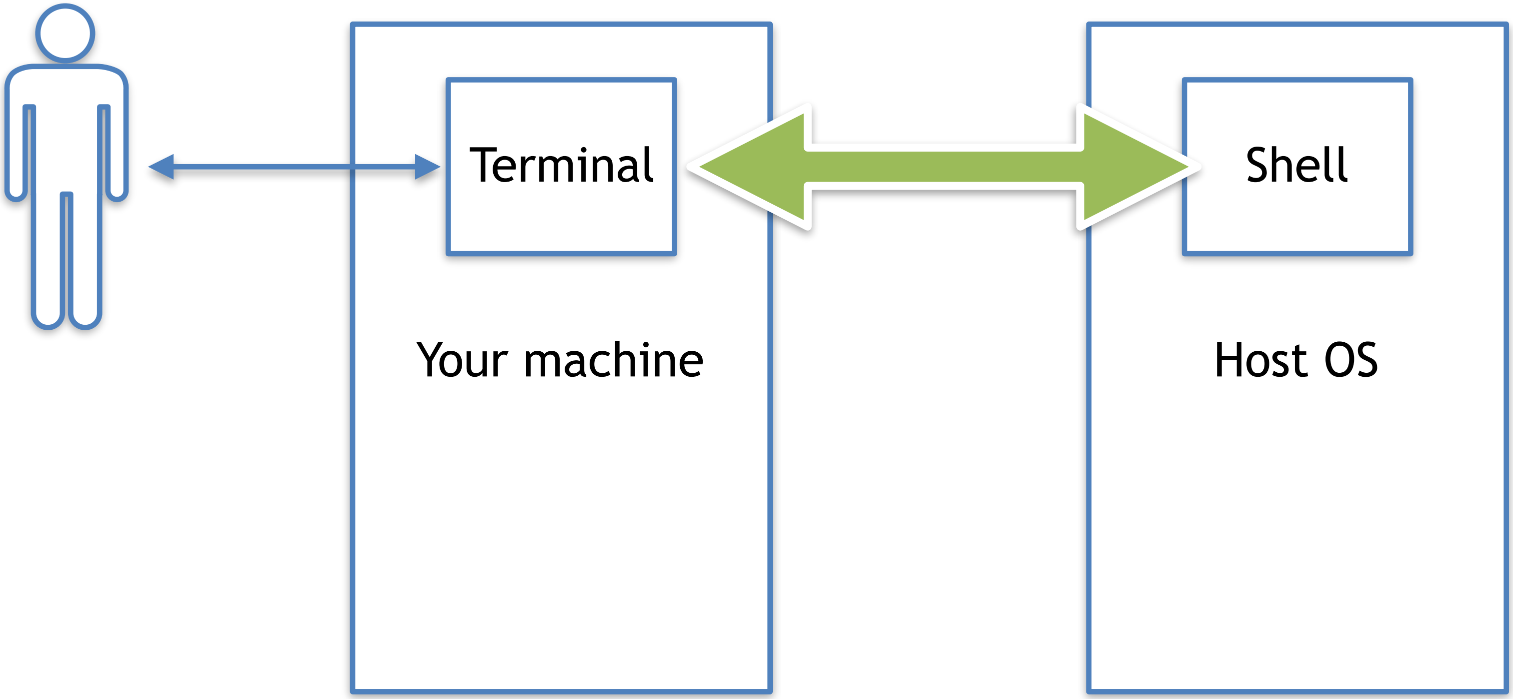
- Shell is a program that takes commands from keyboard and gives them to the operating system (OS).



Working locally



Working remotely



Activity Time!

- Go to **In-Class Exercise 1** on Canvas

If you have time

- On Hamachi
- Use strictly vim, nano, emacs
- Write a python script to compute the sum of first 1000 prime numbers i.e. 2+3+5+…
- Call this script from the command line

Summary

- Now you can ditch GUI completely
  - Maybe?
  - No more annoying mouse clicking :P
- Remotely connect to Hamachi
- Next time:
  - Exploring Linux file system