CS 1217

Lecture 3 – More about processes

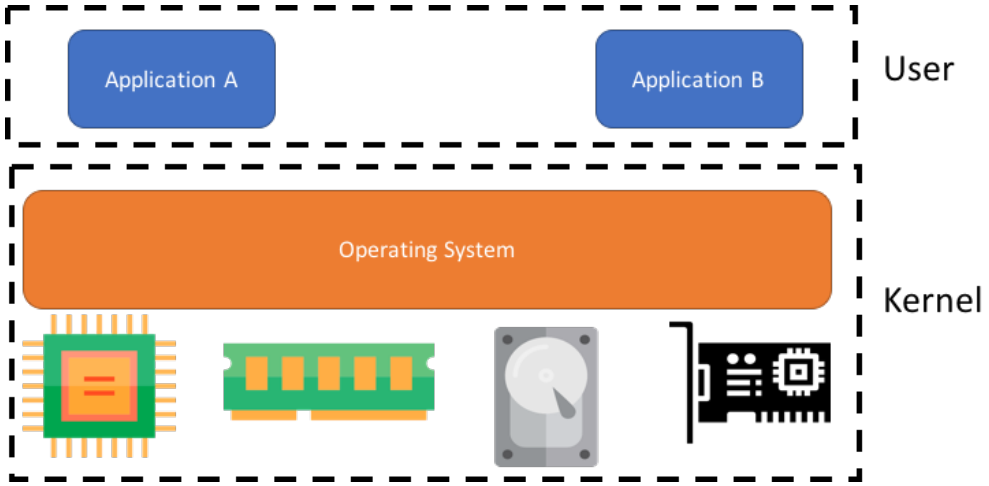
Logistics

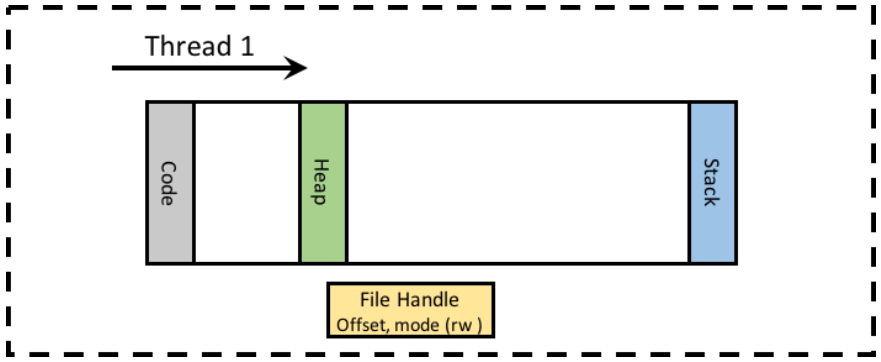
- Assignment 1 goes out today; will be due later this week
- Course cap has been increased

- New lecture rooms from next week
  - Tuesday: AC 01-207 (LT)
  - Wednesday: AC 02 011 (LH)

Recap

- System calls
  - Separation of **privileged** and **unprivileged**
- Process
  - The living entity. As opposed to ____ ? (which is dead)
- The process abstraction
  - Collection of other abstractions
- Inter- (and Intra) process communication mechanisms
  - Shared memory





Processes are Protection boundaries

- OS uses the process abstraction to isolate processes from each other.

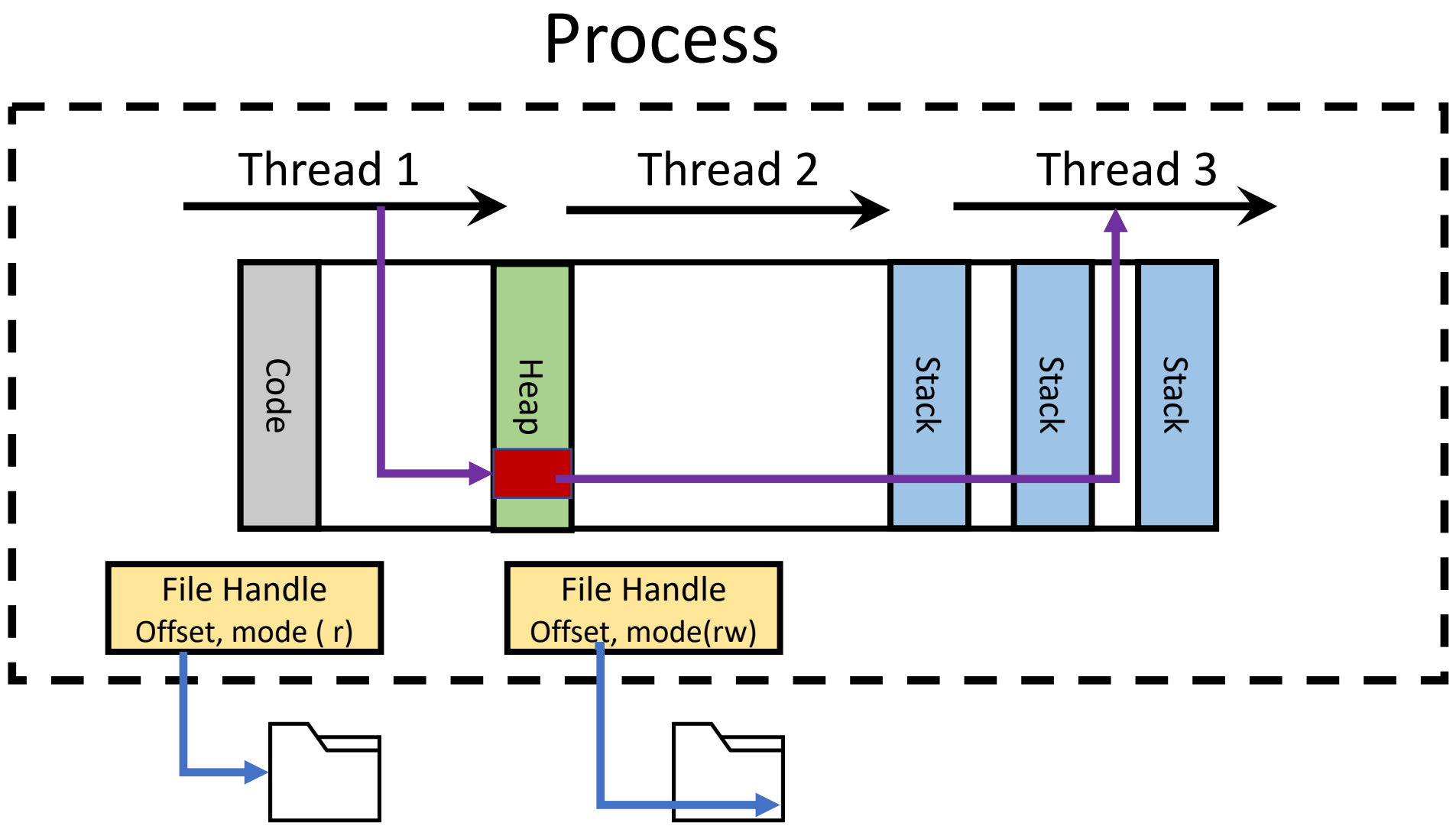
Processes vs. Threads

- Potentially confusing due to overlapping terminology
  - we can describe both a **process** and a **thread** as running.
  - Threads can be thought about as encapsulating CPU state
- Terminology can be helpful for remembering the distinction:
  - A computing **process** requires multiple resources: the CPU, memory, files, etc.
  - A **thread** of execution abstracts CPU state.
- Processes *contain* threads; threads *belong* to a process.
  - Some exceptions, covered later
- A process is considered to be running when one or more of its threads are running.
  - Different operating systems use different terminology, but share common ideas.

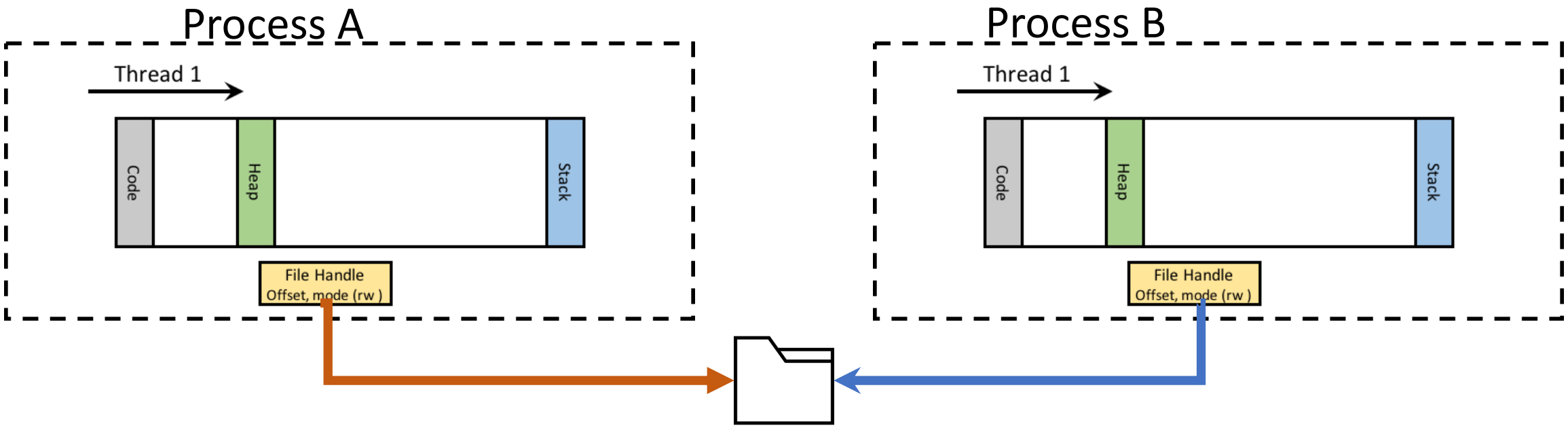
Thread State

- Capturing thread state requires one to capture the CPU state
- CPU state is represented by register contents, both general and special purpose
  - General registers are the architecturally visible registers
  - Special registers are things like PC, Stack and Frame pointers etc.
  - Being able to capture, save, and restore CPU state is useful for things like context switching between processes (more on this later)

Intra - Process Communication



Inter- Process Communication (IPC)



- Simplest way, through shared files
- Other mechanisms: exit codes, pipes, shared memory, signals

Return codes for IPC

```
Manus-MacBook-Pro:code1 manuawasthi$ sleep 10 && /bin/ls &
[1] 820
Manus-MacBook-Pro:code1 manuawasthi$ wait 820
Makefile        a.out           myhello         myhello.c       myhello.h
[1]+  Done                    sleep 10 && /bin/ls
Manus-MacBook-Pro:code1 manuawasthi$ sleep 10 && /bin/test &
[1] 825
Manus-MacBook-Pro:code1 manuawasthi$ wait 825
[1]+  Exit 1                  sleep 10 && /bin/test
```

- Simplest, limited form of IPC.
  - Allows processes to return a **single integer** to the process that created them.
  - 0 indicates success; non-0, failure
- `bash` exposes return codes as `$?`

Pipes for IPC

```
Manus-MacBook-Pro:code1 manuawasthi$ ps au | grep manuawasthi | more
root          347   0.0  0.0  2471532   4192 s000  Us   10:10AM   0:00.03 login -pfl manuawasthi /bin/bash -c exec -la bash /bin/bash
manuawasthi  1155   0.0  0.0  2434840    772 s001  S+   10:38AM   0:00.00 grep manuawasthi
manuawasthi   370   0.0  0.0  2463084   1536 s002  S+   10:10AM   0:00.02 -bash
root          369   0.0  0.0  2471008   4260 s002  Ss   10:10AM   0:00.03 login -pfl manuawasthi /bin/bash -c exec -la bash /bin/bash
manuawasthi   360   0.0  0.0  2463084   1584 s001  S    10:10AM   0:00.24 -bash
root          359   0.0  0.0  2471008   4288 s001  Ss   10:10AM   0:00.03 login -pfl manuawasthi /bin/bash -c exec -la bash /bin/bash
manuawasthi   349   0.0  0.0  2463084   1540 s000  S+   10:10AM   0:00.02 -bash
manuawasthi  1156   0.0  0.0  2434948    620 s001  R+   10:38AM   0:00.00 more
Manus-MacBook-Pro:code1 manuawasthi$
```

- Pipes create a ***producer-consumer buffer*** between two processes.
- Allows output from one process to be used as the input to another.

Signals

- Signals are a form of asynchronous communication between processes.
- Processes can register a **signal handler** to run when a signal is received.
- Users can send signals to processes owned by them; the super-user can send a signal to any process.
- Processes can ignore signals
  - SIGKILL is a notable exception; used for non-graceful termination.
  - SIGTERM is used for graceful shutdown.

Signals

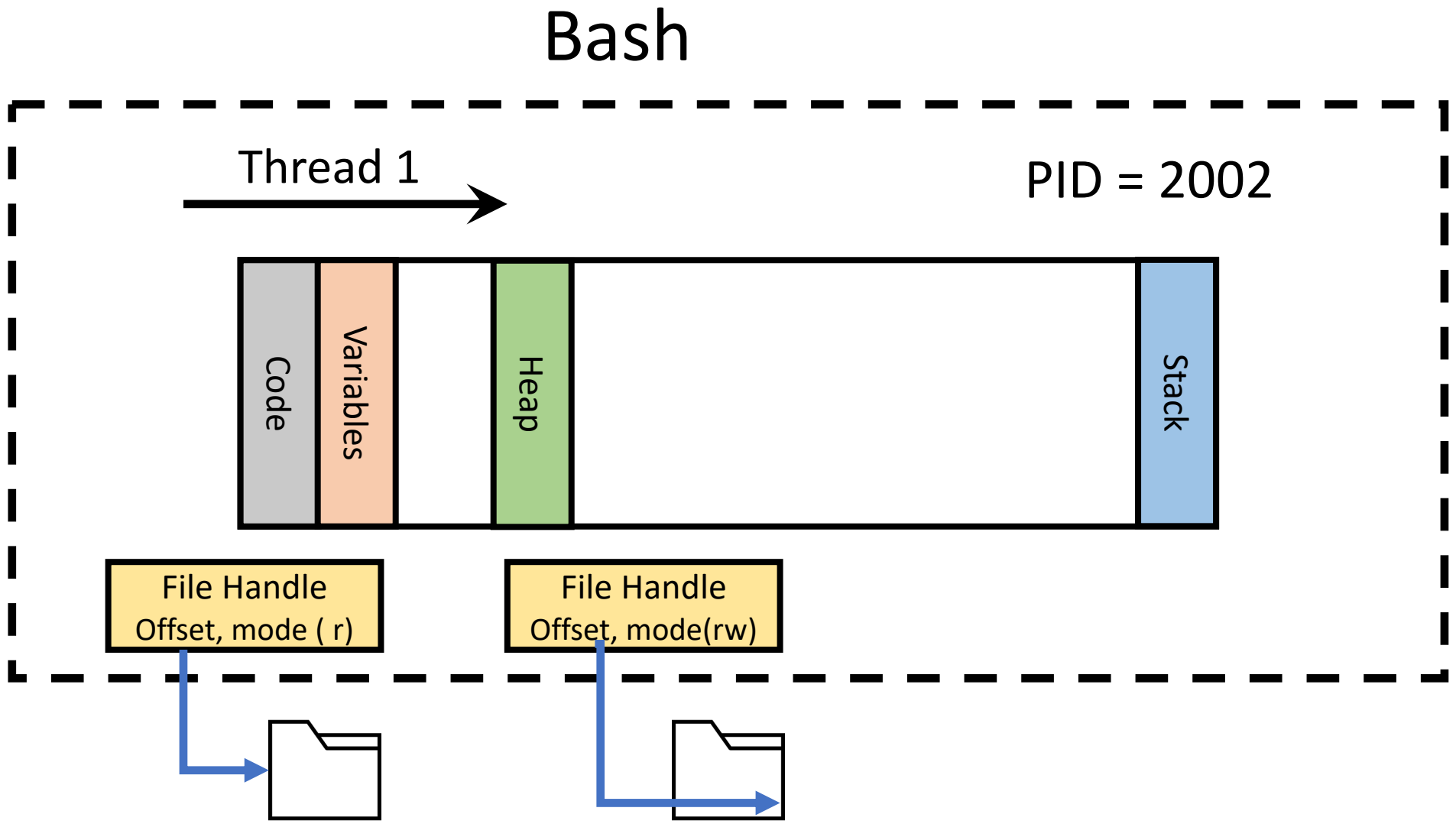
```
cs304@cs304-devel:~$ sleep 1200 &
[1] 3392
cs304@cs304-devel:~$ kill 3392
cs304@cs304-devel:~$
[1]+  Terminated              sleep 1200
cs304@cs304-devel:~$ sleep 1200 &
[1] 3406
cs304@cs304-devel:~$ kill -9 3406
cs304@cs304-devel:~$
[1]+  Killed                  sleep 1200
cs304@cs304-devel:~$
```

```
Some of the more commonly used signals:
1       HUP (hang up)
2       INT (interrupt)
3       QUIT (quit)
6       ABRT (abort)
9       KILL (non-catchable, non-ignorable kill)
14      ALRM (alarm clock)
15      TERM (software termination signal)
```

Processes: Example Chrome

- **Chrome has multiple threads. What are they doing?**
  - Waiting for and processing interface events: mouse clicks, keyboard input, etc.
  - Redrawing the screen as necessary in response to user input, web page loading, etc.
  - Loading web pages—potentially *in parallel* to speed things up. (Partial list)
- **Chrome is using memory. For what?**
  - the executable code of Chrome itself.
  - Shared libraries for web page parsing, security, etc.
  - Stack(s) and heaps
- **Chrome has files open.**
  - stderr, stdout etc
  - Configuration files.
  - Fonts

The Process Abstraction



cpuinfo and meminfo

```
cs304@cs304-devel:~$ cat /proc/cpuinfo | more
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 61
model name      : Intel(R) Core(TM) i5-5257U CPU @ 2.70GHz
stepping        : 4
cpu MHz         : 2699.998
cache size      : 3072 KB
physical id     : 0
siblings        : 1
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid  : 0
fpu             : yes
fpu_exception   : yes
cpuid level     : 20
wp              : yes
```

```
cs304@cs304-devel:~$ cat /proc/meminfo
MemTotal:        2041304 kB
MemFree:          193196 kB
MemAvailable:     925720 kB
Buffers:           54232 kB
Cached:           790040 kB
SwapCached:            0 kB
Active:          1335996 kB
Inactive:         354308 kB
Active(anon):     846872 kB
Inactive(anon):     5368 kB
Active(file):     489124 kB
Inactive(file):   348940 kB
Unevictable:          16 kB
Mlocked:              16 kB
SwapTotal:        728520 kB
SwapFree:         728520 kB
Dirty:                92 kB
Writeback:             0 kB
AnonPages:        846056 kB
Mapped:           225512 kB
Shmem:              6212 kB
Slab:              83600 kB
SReclaimable:      52028 kB
SUnreclaim:        31572 kB
KernelStack:        7820 kB
PageTables:        38140 kB
NFS_Unstable:          0 kB
```

Process Information: Bash

- What is `bash`?
- How do I figure out its process information?

```
cs304@cs304-devel:~$ ps
  PID TTY          TIME CMD
 2002 pts/1    00:00:00 bash
 2078 pts/1    00:00:00 ps
cs304@cs304-devel:~$ pgrep bash
2002
cs304@cs304-devel:~$ ps aux | grep bash
cs304     2002  0.0  0.2  29684  5000 pts/1    Ss   07:42   0:00 bash
cs304     2084  0.0  0.0  21536  1108 pts/1    S+   07:47   0:00 grep --color=auto bash
cs304@cs304-devel:~$
```

Process Information: Bash

- What has changed here?

```
cs304@cs304-devel:~$ ps
  PID TTY          TIME CMD
 2002 pts/1    00:00:00 bash
 2010 pts/1    00:00:00 ps
cs304@cs304-devel:~$ pgrep bash
1848
2002
cs304@cs304-devel:~$ ps aux | grep bash
cs304     1848  0.0  0.2  29816  5088 pts/0    Ss+  07:37   0:00 bash
cs304     2002  0.1  0.2  29684  5000 pts/1    Ss   07:42   0:00 bash
cs304     2016  0.0  0.0  21536  1032 pts/1    S+   07:42   0:00 grep --color=auto bash
cs304@cs304-devel:~$
```

Detailed Process information

```
cs304@cs304-devel:~$ ps -lF 2002
F S UID          PID  PPID  C PRI  NI ADDR SZ WCHAN    RSS PSR STIME TTY          TIME CMD
0 S cs304       2002  1838  0  80   0 -  7454 wait    5284   0 07:42 pts/1    0:00 bash
cs304@cs304-devel:~$
```

- UID
  - user the process is running as
- PID
  - Process ID
- PPID
  - Parent Process ID
- SZ
  - Size of core process image
- WCHAN
  - address of the kernel function where the process is sleeping
- RSS
  - resident set size, amt of memory used by process
- PSR
  - Processor
- STIME
  - Start time
- TIME
  - Time process has been running
- CMD
  - Command used to start process

pmap

- Information about memory mappings of process

```
cs304@cs304-devel:~$ pmap 2002
2002:   bash
0000560445ea5000   1040K r-x-- bash
00005604461a8000     16K r---- bash
00005604461ac000     36K rw--- bash
00005604461b5000     40K rw---   [ anon ]
00005604480ff000   1604K rw---   [ anon ]
00007fb4b8a15000   1948K r-x-- libc-2.27.so
00007fb4b8bfc000   2048K ----- libc-2.27.so
00007fb4b8dfc000     16K r---- libc-2.27.so
00007fb4b8e00000      8K rw--- libc-2.27.so
00007fff083df000    132K rw---   [ stack ]
```