

So far

Introduction

~~Software security~~

Host security

Network security

Web security

Today

Processes

Users

Next time:

Authorization

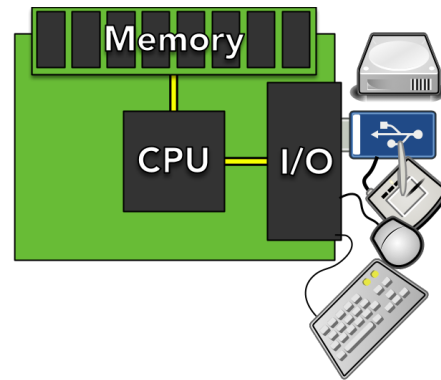
Processes

A process is a *running program*

Processes have memory

Processes execute

Processes have indirect access to I/O



4 / 14

We've been working with processes already, but we haven't really defined them. A process is just a _____, and it runs on a _____ computer. To see the processes that are currently running on your computer, you can use a GUI like Activity Monitor or Task Manager, or you can use the `ps` command at the command line.

Run `ps aux | wc -l` on macOS, FreeBSD.

The addresses used directly by your software are _____. This is why, when the same code is executed in ten processes in parallel, they can all refer to the _____, but *without interference*.

The OS must *schedule* processes for execution, deciding which process will get to run on which processor at which time. This is a topic for ECE 8400 / ENGI 9875.

External resources such as data on a disk or a network connection are generally/typically/almost always represented as _____.

Files

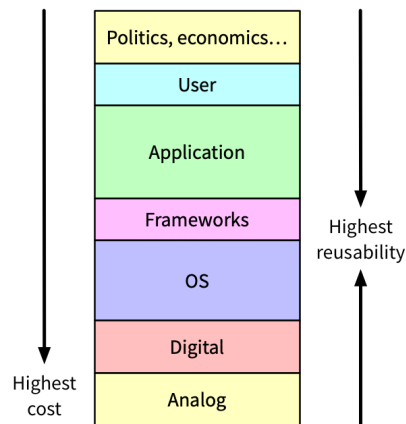
Multiple abstraction layers:

```
new PrintWriter(new BufferedWriter(new File
```

```
std::cout << "Hello, world!" << std::endl;
```

```
fwrite(stdout, "Hello, world!\n");
```

```
char message[] = "Hello, world!\n";  
int fd = STDOUT_FILENO;  
write(fd, message, sizeof(message));
```



5 / 14

There are lots of ways to look at files from lots of levels of abstraction. Each layer tends to add something nice and/or helpful, but at bottom, all of these layers' concepts of files are rooted in an _____ . This is because no library can control a disk: only _____ can.

This is another example of how some interfaces are more "real" than others. Interfaces within a single address space — such as library APIs — can wrap up the functionality of lower-level libraries, but when it comes down to the level of _____ (as in a _____ programming language), there is no "real" separation among them. However, interfaces between different _____ of things (users and computers, processes and the OS, software and hardware) tend to be much clearer, "harder" interfaces.

Process file abstractions

Each process has a set of integer *file descriptors*

Can use *system calls* to open, close, read, write, etc.

```
int fd = open("/home/jon/hello.txt", O_RDONLY);  
  
/* ... */  
  
write(fd, some_data_bytes, data_length);  
  
/* ... */
```

6 / 14

A file descriptor can then get wrapped up, together with some buffering for performance, into a _____ . That can then get wrapped up into a C++ _____ , which might be used to implement a Java `java.io.FileWriter`.

File I/O system calls

From a process' perspective:

- system calls are C functions
- files are named by small integer indices (e.g., FD 3)
- each process has its **own** array of files
- ... and that's enough detail for now

7 / 14

There's a lot more detail to be dug into about file descriptors (e.g., how `libc` communicates with the OS kernel), and we'll go into that detail in ECE 8400 / ENGI 9875. In fact, our first lab in that course will have you invoking system calls via nothing but native instructions in assembly code! For now, however, this is all that we need to know in order to start talking about host security.

Processes

Processes have *memory* – virtual memory

Processes *execute* – threads

Processes have (indirect) access to *resources* – files

Processes execute on behalf of... ???

Users

Username

User IDs

User authentication

User authorization

9 / 14

What's a user? A human being, sure, but how does the computer see a user? How do we think of users?

We often think of users as being identified by _____: short, human-readable names that are unique to _____. If I'm using a computer, I can see my current username by running `whoami(1)`.

Something a bit more meaningful to the computer, however, is not a _____ but a _____. User IDs are still short and _____, but instead of strings, they're _____. You can see information about your user ID (and group IDs!) by running `id(1)` (or just `whoami` on Windows).

How does the computer know that the person sitting in front of it is *actually* the person they claim to be? That's called user *authentication*, and we'll get into it when we start talking about passwords (and their problems, and alternatives).

Next time, we'll start getting into user *authorization*: saying _____ is allowed to do _____ to _____.

User databases

Where is user information stored?

- Active Directory
- Binary databases (e.g., Berkeley DB caches)
- NIS[plus] (though not really any more)
- OpenLDAP
- Text files (e.g., `/etc/passwd`)

10 / 14

We'll examine user details specific to discretionary access control next time, but for now we will look at the contents of files like `/etc/nsswitch.conf`. This *Name Service Switch* config file tells a Unix machine where to find information about lots of kinds of names: users, groups, protocols, shells, etc. It can point us at `files`, at binary `db` files (typically used as a cache), at `ldap` (which might actually be Active Directory) or at the basically-defunct `nisplus`.

User database: files

```
# $FreeBSD$
#
root:*:0:0:Charlie &:/root:/bin/sh
toor:*:0:0:Bourne-again Superuser:/root:
daemon:*:1:1:Owner of many system processes:/root:/usr/sbin/nologin
operator:*:2:5:System &:/usr/sbin/nologin
bin:*:3:7:Binaries Commands and Source:/usr/sbin/nologin
tty:*:4:65533:Tty Sandbox:/usr/sbin/nologin
kmem:*:5:65533:KMem Sandbox:/usr/sbin/nologin
games:*:7:13:Games pseudo-user:/usr/sbin/nologin
news:*:8:8:News Subsystem:/usr/sbin/nologin
man:*:9:9:Mister Man Pages:/usr/share/man:/usr/sbin/nologin
sshd:*:22:22:Secure Shell Daemon:/var/empty:/usr/sbin/nologin
# ...
```

11/14

We can examine local user details via:

```
cat /etc/passwd
```

If we look at `/etc/passwd` on a local Unix-like machine, we should see users like the user who set up the box. On a LabNet machine, however, _____...

User database: LDAP

Lightweight Directory Access Protocol

Queries *directory servers*:

- Active Directory
- ApacheDS
- FreeIPA / 389 directory server
- OpenLDAP

12 / 14

In practice, Microsoft Active Directory is absolutely dominant, as most large networks support large numbers of Windows PCs.

In a LabNet environment, we can query lots of interesting details from LDAP using commands like:

```
ldapsearch -H ldaps://dogbert.cs.mun.ca "(uid=p15jra)"
```