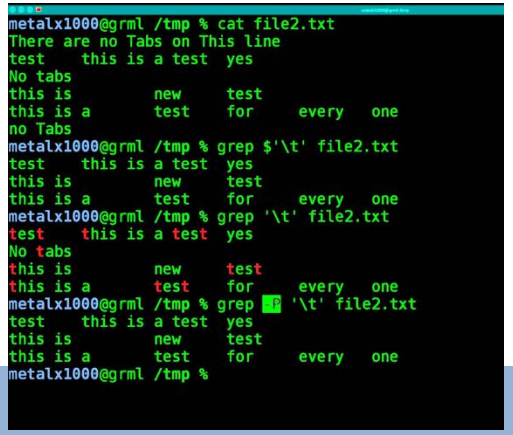
Operating Systems

Eric Lo

1. Overview of an Operating System

What is an OS?

- An OS
  - Includes a **program**
    - called "**kernel**" (e.g., kernel.exe)
      - which manages all the physical devices (e.g., CPU, RAM and hard disk)
      - exposes some functions as **system calls** for others to configure the kernel or build things (e.g., C library) on top
  - Includes some more **programs**
    - called "**drivers**"
      - which handles the interaction between the kernel and the external devices (e.g., keyboard)
    - called a "**shell**"
      - Which renders a simple command-line user interface with a full set of commands
    - …
  - Includes some "optional" **programs**
    - GUI, Browser, Paintbrush, …



What is a process?

- A process is an execution instance of a program.
  - More than one process can execute the same program code

Let's consider the following two commands

Command A	`ls -R /`	Recursively print the directory entries, starting from the directory '/'
Command B	`ls -R /home`	Recursively print the directory entries, starting from the directory '/home'

They are 2 different processes

A process is more than a program

- A process has states concerning the execution.
  E.g.,
  - which line of codes it is running;
  - how much time left before returning the CPU to the others

- Commands about processes
  - e.g., **`ps`** & **`top`**.

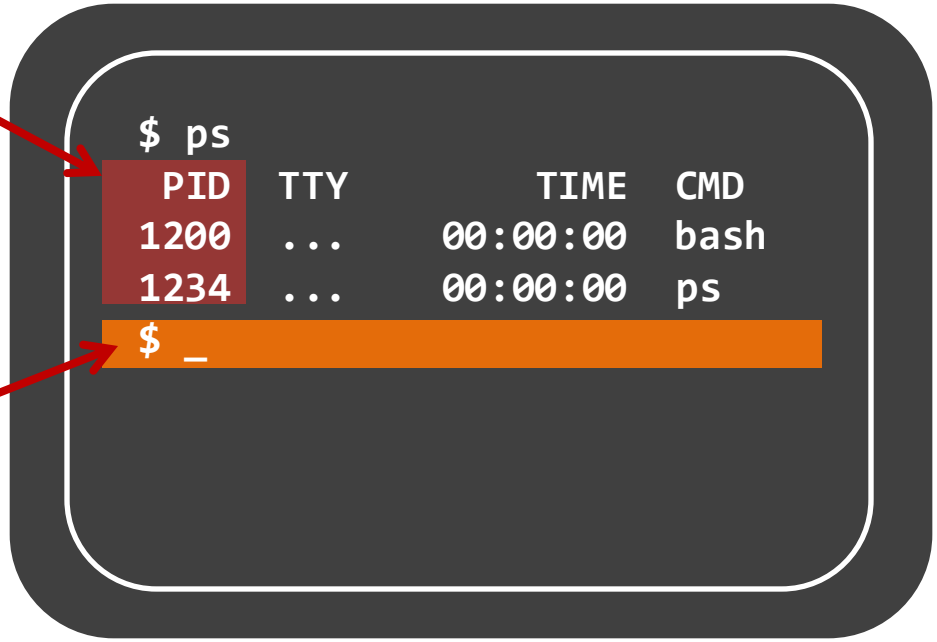
Process-Related Tools

- The tool "**ps**" can report a vast amount of information about every process in the system
  - Try "**ps -ef**".

This column shows the unique identification number of a process, called **Process ID**, or PID for short.

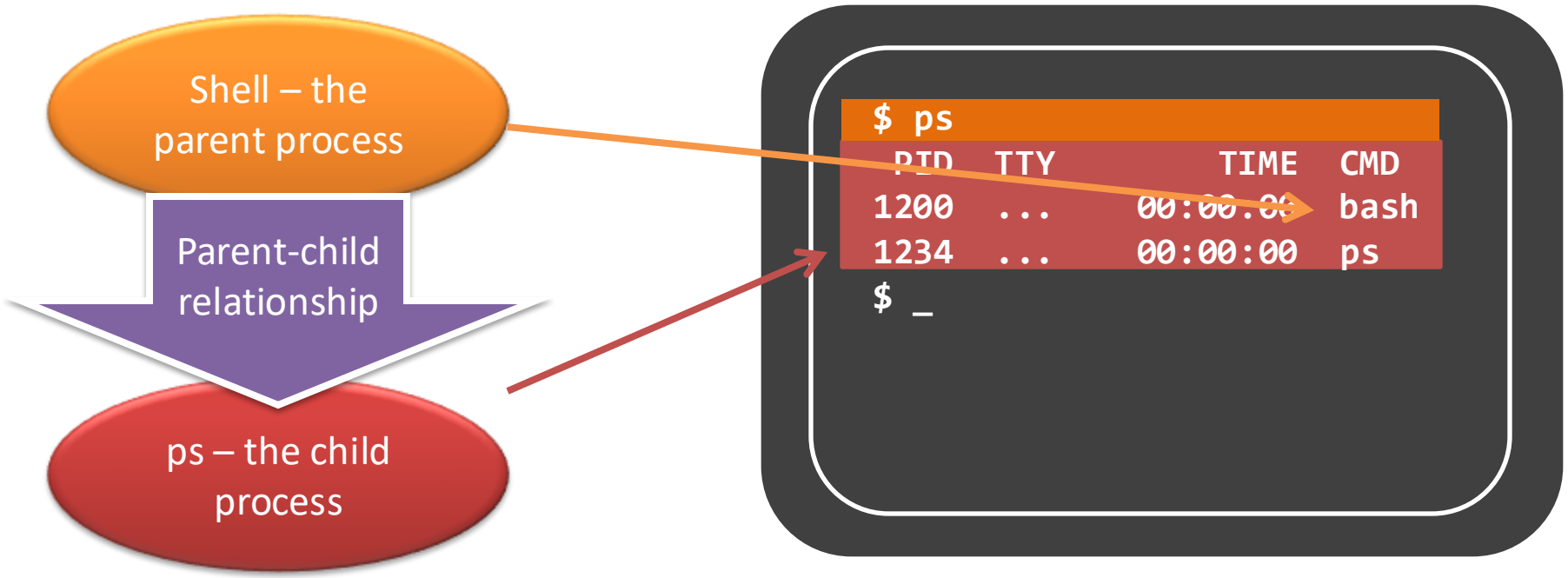
Hint: you can treat **ps** as the short-form of "**process status**"

By the way, this is called **shell**.



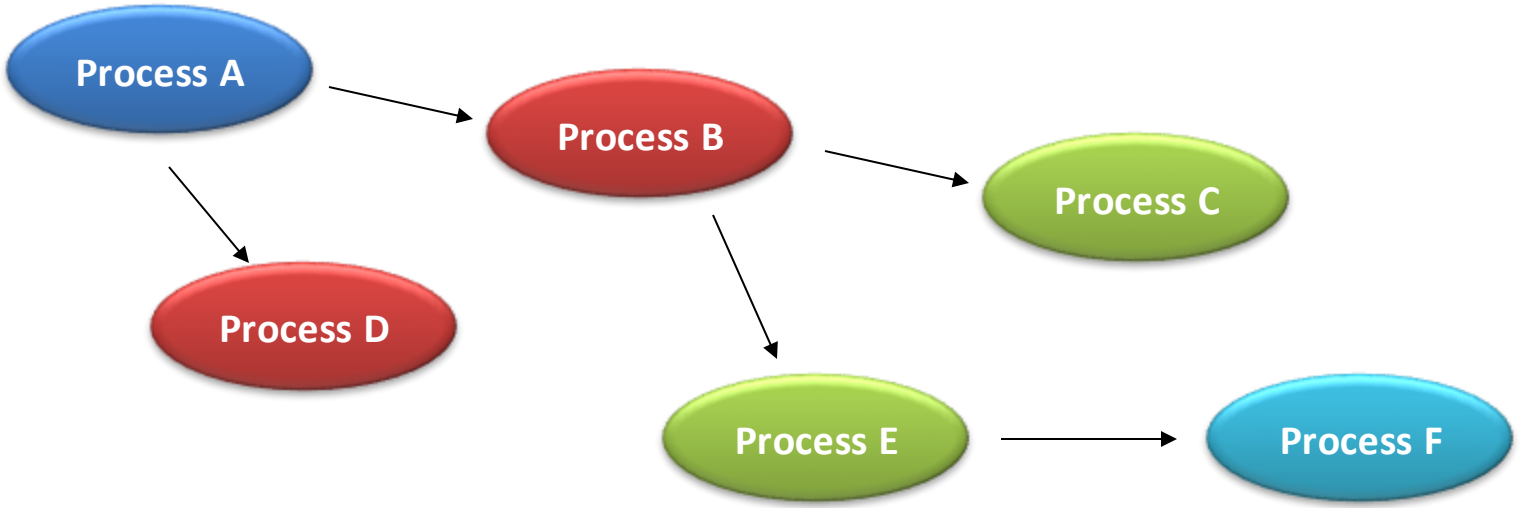
What is a Shell?

- A shell is a program
- You open a “terminal”, which actually launches a “shell” process
  - E.g., bash in Linux
- Written in C
    - use `getchar()` (to get your command “**ps**”)
    - syntax checking
    - invoke a function `fork()` (a **system call**) to create a new process
      - i.e., becoming a **child process** of the shell.
    - Ask the the child process to `exec()` the program “**ps**”.



Process hierarchy

- Process relationship:
  - A parent process will have its child processes.
  - Also, a child process will have its child processes.
  - This forms a **tree hierarchy**.



E.g., “Process E” is the shell and “Process F” is “**ps**”.

What is a system call?

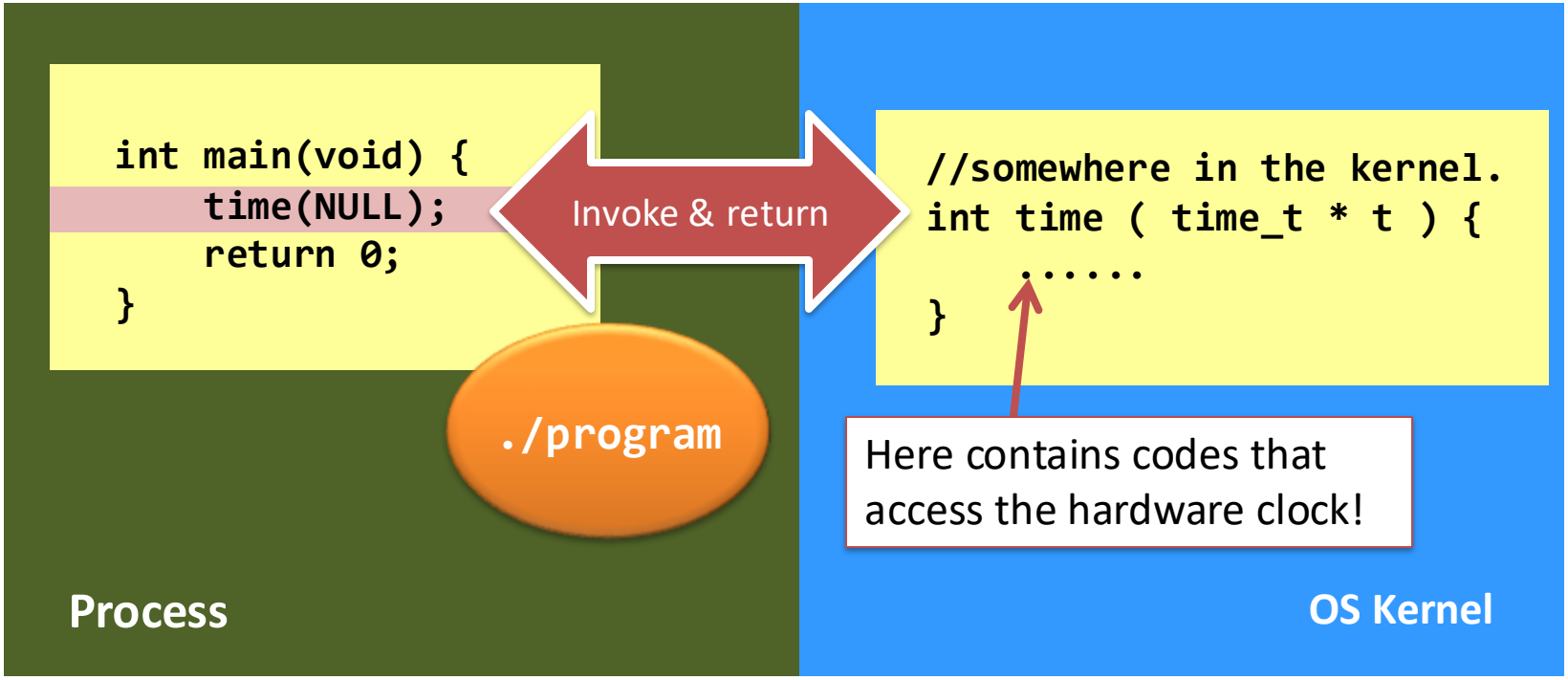
- System call
  - Is a function call.
  - Exposed by the **kernel**.
  - Abstract away most low-level details.
    - Do you know how to read an input from keyboard?

```
int add_function(int a, int b) {
    return (a + b);
}

int main(void) {
   int result;
   result = add_function(a,b);
   return 0;
}

// this is a dummy example…
```

Function implementation.

This is a function call.

Interacting with the OS



System calls

- Categorizing system calls as follows:

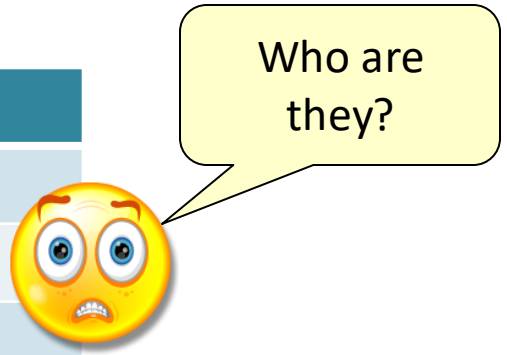
Process	File System	Memory
Security	Device	Information

System calls

- How can we know if a “function” is a system call?
  - Read the man page “**syscalls**” under Linux.

- Without reading the man pages, guess which of the following is/are system call(s)?

Name	Yes/No?
`printf() & scanf()`	No
`malloc() & free()`	No
`fopen() & fclose()`	No
`mkdir() & rmdir()`	Yes
`chown() & chmod()`	Yes

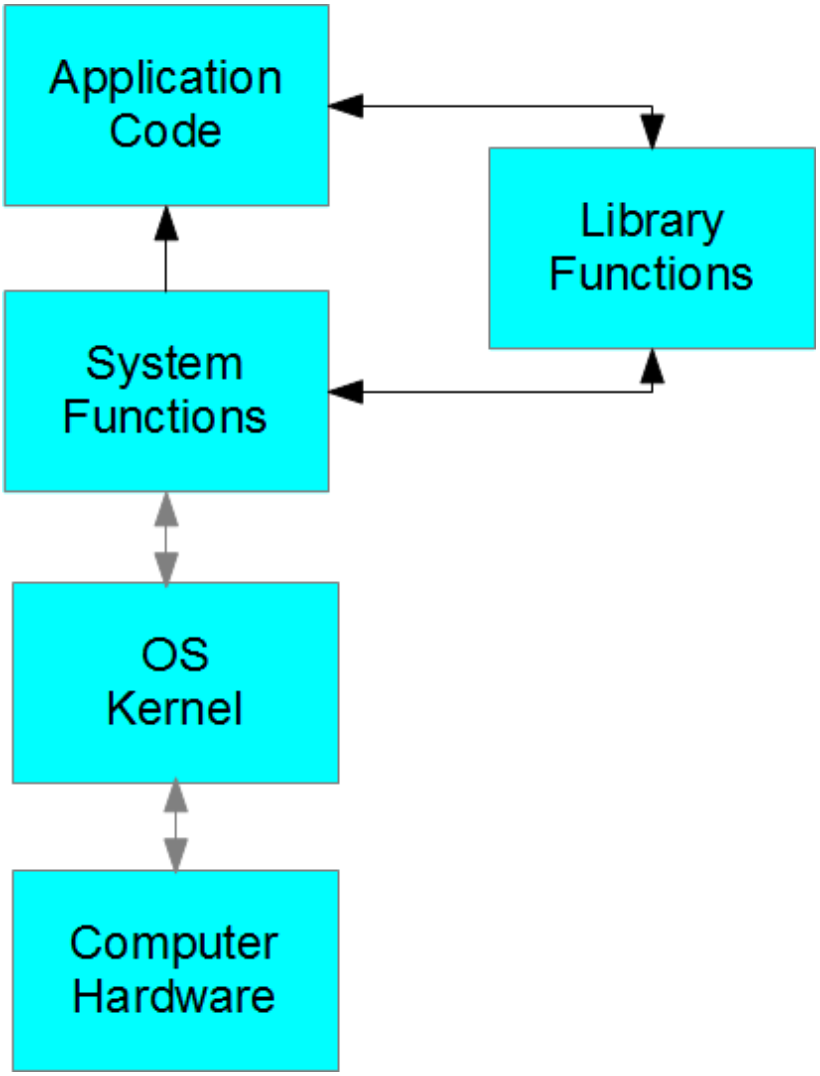


System calls VS Library function calls

- System call vs **Library call**
- Take **`fopen()`** as an example.
  - **`fopen()`** invokes the system call **`open()`**.
  - So, why people invented **`fopen()`**?
  - Because **`open()`** is too primitive and is not programmer-friendly!

Library call	`fopen("hello.txt", "w");`
System call	`open("hello.txt", O_WRONLY | O_CREAT | O_TRUNC, 0666);`

System calls VS Library function calls



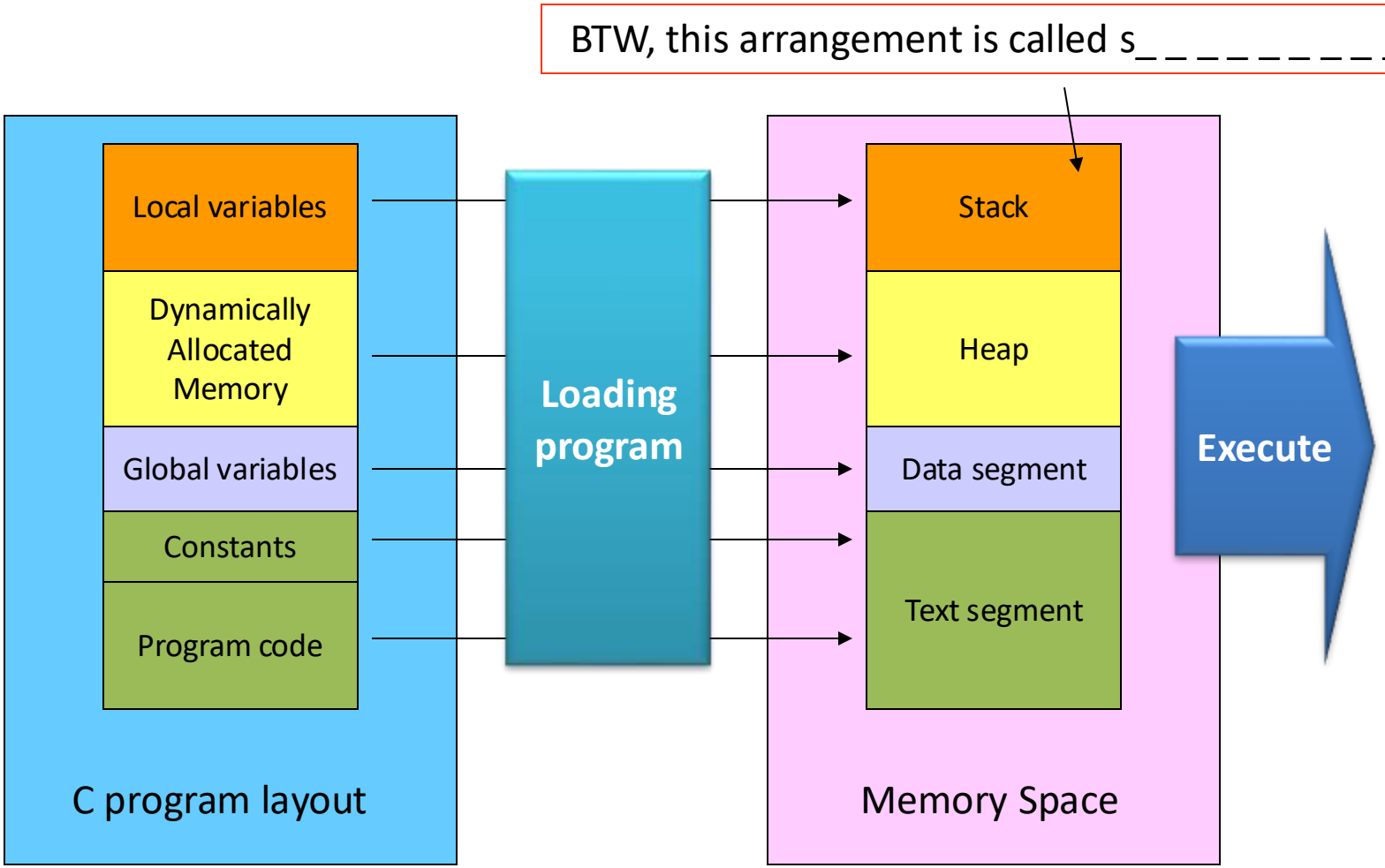
What will we learn about processes?

- System calls
  - How to program a simple, bare-bone shell?
- Lifecycle and Scheduling
  - How to create processes?
  - How to handle the death of the processes?
  - Which process shall get the core next?
- Signals
  - How to suspend a process?
  - A virus? We can make a program to play a song whenever you type **Ctrl+C**?
- Synchronization
  - How processes can cooperate to do useful work together?

Introduction to Operating System Components

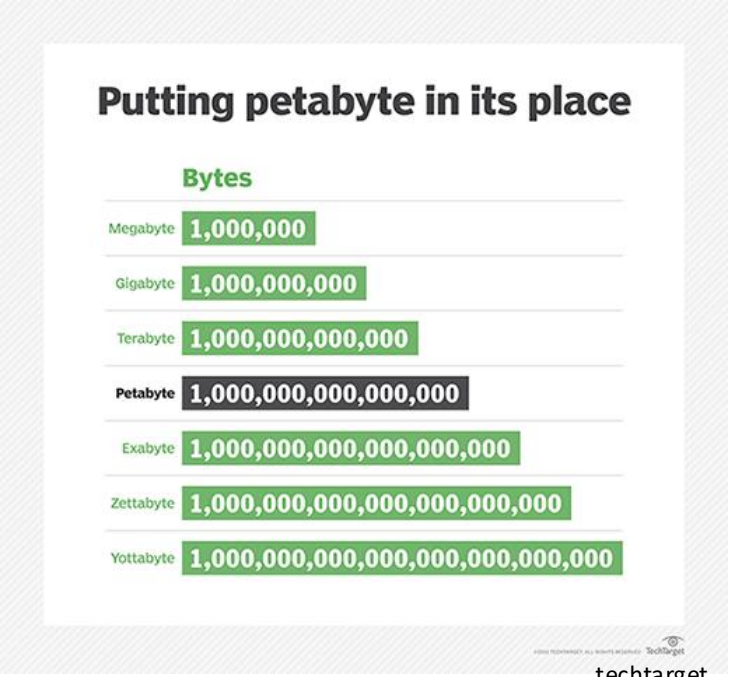
Memory

Process' Memory



What will we learn about memory?

- Virtual memory
  - 1 process virtually owns **all**
    - 2^{32} RAM (=4GB) for 32-bit CPU
    - 2^{64} RAM (=16EB) for 64-bit CPU
      - Practically OS set it to be 256TB or 1PB
- Memory-related functions
  - E.g., how to write "**`malloc()`**"?
- Stack overflow
  - Why & when?
- RAM = 256MB
  - **`malloc(16MB)`**
  - How much free memory left?



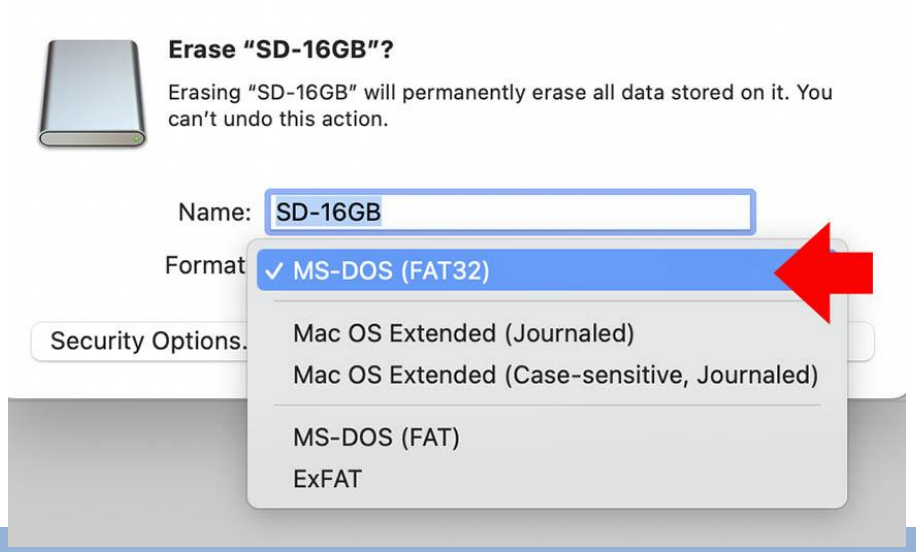
techtarget

Introduction to Operating System Components

File System

What is a File System?

- Have you heard of…
  - FAT16, FAT32, NTFS, Ext3, Ext4, BtrFS, Juliet?
  - They are all file systems.
  - It is about how to organize your files in the storage device.



What is a File System?

- If a FS just lays your files one-by-one, consecutively, tightly, in your hard disk, is it good?
  - What if you increase the size of your file?
  - What's the performance of searching for a file? O(?)
  - BTW, how to deal with directories?

What we will learn about FS?

- How to deal with directories?
- Implementation of some famous FS-es.
- Why does a file system perform badly?
- How to undelete a file?

More… from System Programming to Programming Operating System

- Multi-threading
- Booting
- Architectural Conscious OS Programming
- Lock-free Programming
- I/O
- Virtualization