SY110
OS - Windows Shell and Permissions

Major Brian Hawkins, USMC

U.S. Naval Academy

Fall AY 2018

Outline

Quick Review

Operating System access methods

What are they?

- The **GUI**
- The **Shell**
- The **API**

Quick Review

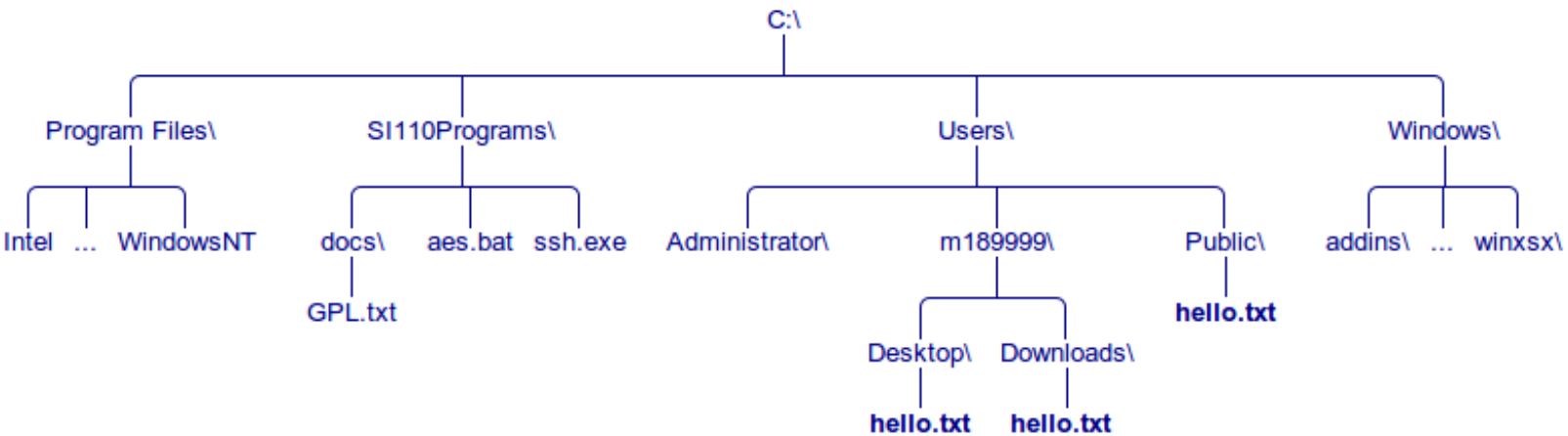
Operating System access methods

What they are

- The **GUI** – Graphical User Interface. Point and click environment; for users.
- The **Shell** – An interface that allows commands to be entered as plaintext strings – on Windows, called the *Command Prompt*; for users and programs.
- The **API** – Application Programming Interface. A direct method for programs to ask the OS to perform tasks on their behalf; only for programs.

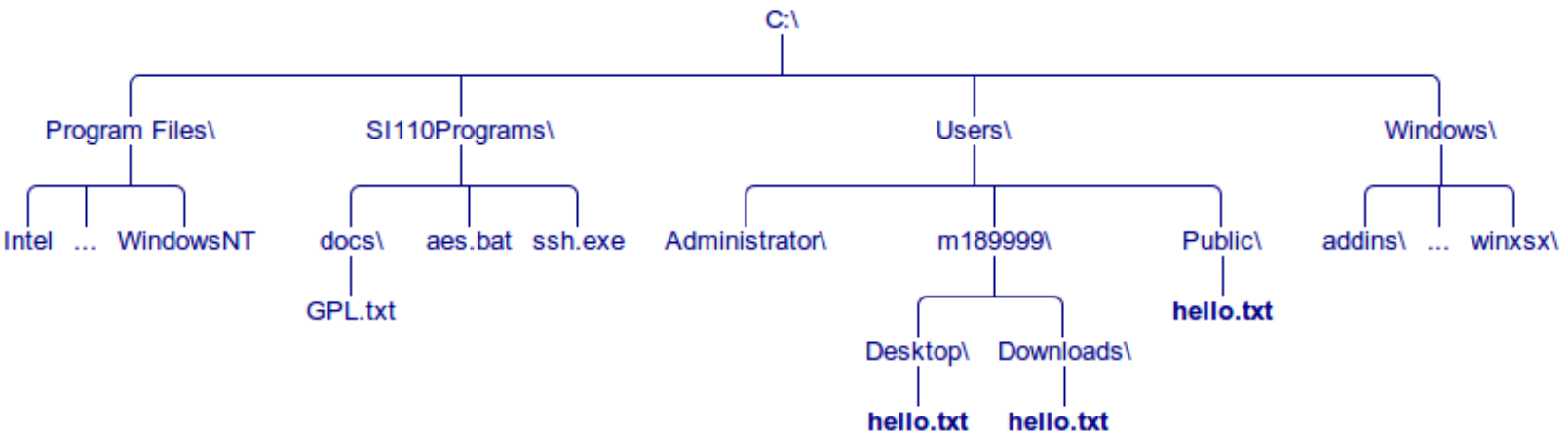
File systems

In general, what is a file system?



File systems

A hierarchical organization of files/folders within your computer. The structure and conventions may be different between systems, however. For example, Windows and Unix-based OSes use different file systems.



What is the **root** of this file system? How do we know this is a Windows file system? What are the **paths** to the three `hello.txt` files?

Programs and Processes

What is a process?

A *process* is simply an executing instance of a Program (and a Program is just a file). Recall that a Program is a file on your hard drive. When we run this program, its bytes get copied into RAM, and executed by the CPU. With some other bookkeeping information, this executing copy of the Program is called a process. Multiple instances of the same process can be running at the same time. The OS monitors all processes status, resources they're using (RAM, CPU load, etc) and privileges.

User accounts, logins, permissions

Every running process has a username that is attached to it – this is the name of the user that caused the Program to be run.
Similarly, every file and directory also has a username attached to it – referred to as the file or directory's owner.

Usually, the OS will deny any attempt to manipulate a file or folder unless the process owner and owner of the file or directory match. This is because each file/folder has a set of permissions – rules that specify which users can take what actions on them.

Permissions are what make the system secure – ensuring that only users that have the right credentials can launch processes that access files on the system. Thus, managing system logins is important!

User accounts, login, permissions

If I log into your computer with my login name, the OS should not allow me to launch a process that can read, write to, and delete files that belong to you!

The exception

There is usually one account on a computer that has unlimited privileges – called the "Administrator" account on Windows, or "root" on Unix-based systems. Sometimes we need to launch a process with Administrator or root privileges because it needs to access special areas of the file system or do things that process normally could not (*e.g.* installing new software needs access to special parts of the file system

Security implications

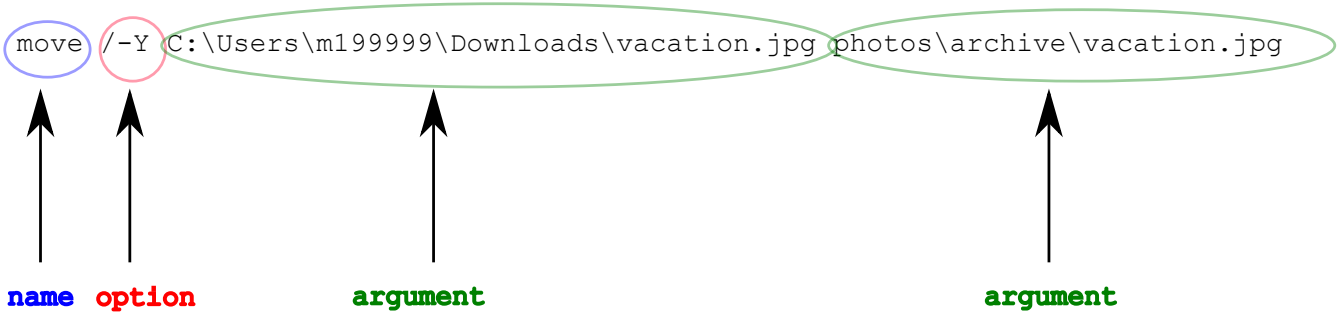
What happens if I can access the Administrator/root account on someone else's machine?

Shell commands

Anatomy of a shell command

A shell command consists of:

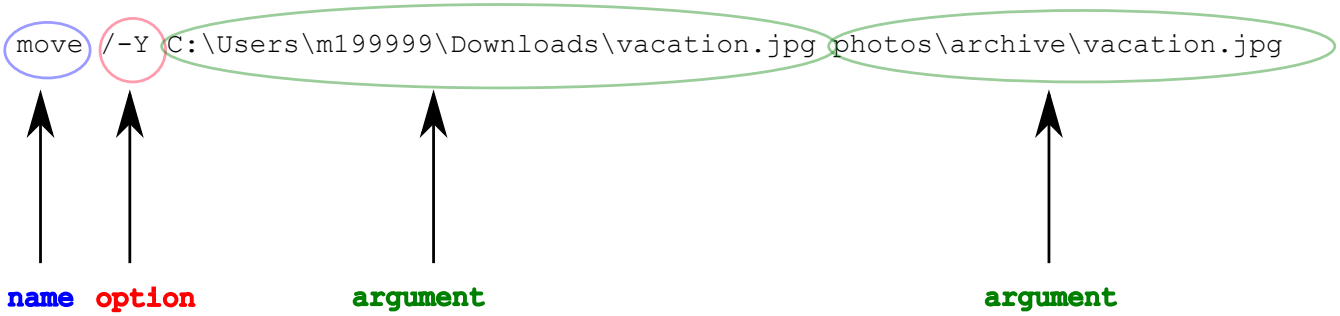
- A command *name*
- Zero or more *options*
- Zero or more *arguments*



Shell commands

Anatomy of a shell command

- Command name – Name of the program to have the shell execute
- Arguments – Information the command needs to execute, *e.g.* the path to a file to move and where to move it
- Options – Special arguments that modify the program's behavior. Start with a "/" or "-" character in Windows, just "-" in Unix-based systems



Shell commands

There are tons and tons of shell commands, but you should know these command names, what they do, and what arguments they expect:

- `dir` – list the contents of the current directory (folder)
- `cd arg1` – change directory to the "arg1" directory
- `mkdir arg1` – make the directory (folder) called "arg1"
- `rmdir arg1` – delete the directory called "arg1"
- `del arg1` – delete the file called "arg1"
- `copy arg1 arg2` – copy the file "arg1" to a file called "arg2"
- `move arg1 arg2` – move (rename) the file "arg1" to a file called "arg2"
- `type argument1` – Prints bytes of the file "arg1" interpreted as ASCII characters

Identify the components

```
copy foo\bar.txt .  /V
putty -telnet james@brown.soul.org
```

- What are the command names?
- What the arguments?
- Options?

A quick gotcha – what does `mkdir foo bar` do? What if we want to *one* directory called `foo bar` though? Answer – enclose the arguments with quotes: `mkdir ‘‘foo bar’’`

Shell command exercises

Let's get some hands-on experience to make using the shell more concrete

Questions

Questions?