From Program to Process

- Topics
  - Managing and constructing processes.
- Learning Objectives:
  - Explain the difference between a program and a process.
  - Identify the different pieces that the operating system needs to keep track of for each process.

Managing the Wily Process

- The operating system is tasked with keeping track of all the processes executing (or attempting to execute) on a system.
- The OS would like to keep track of as little information as possible, but it must keep track of enough to facilitate execution.
- The OS also keeps track of enough information to make intelligent resource allocation decisions.
- So, the OS keeps track of various things for each process:
  - Execution state(s)
  - Resource Information
  - The memory comprising the process' address space

Programs to Processes

- A program is a simple executable file.
- A process is an entity able to run on a processor.
- What are the basic pieces of a process?
  - Parts visible in the executable
  - Parts that only happen when you run or get ready to execute the process

The Parts of the Executable (1)

- An executable [file] is constructed from a set of [compiled] object modules and libraries.
- Linking is the process of constructing an executable.
  - Static linking: the executable contains all the code and data that will be needed to run the program.
  - Dynamic linking: the executable contains references to modules that will be accessed at load time.

The Parts of an Executable (2)

- Code: the instructions that are executed when this runs; typically read-only.
- Rdata: read-only data
  - Constant strings
  - Jump [dispatch] tables
- Data: initialized global variables
- Bss: “Block Started by Symbol”
  - Uninitialized global variables

Organizing your executable

- ELF is a commonly used format for executables.
- ELF = Executable and Linkable Format.
- If you have never learned about linking, view the linking video on the web site.
- Whether you have or not, you will find it useful to examine the the OS/161 files that deal with elf headers:
  - `kern/include/elf.h`
  - `kern/syscall/loadelf.c`
- The function `load_elf` will read an elf executable into an address space; you might find this useful when you are exec-ing processes.

Parts of a Process not in the Executable

- Execution State
- Resources
- Address space
- Miscellaneous

Execution State

- If the process is currently running, the execution state is captured in the hardware:
  - Register contents
  - Program counter
  - Stack pointer
  - Etc.
- If the process is **not** running, then the OS needs to have a snapshot of this information.
  - PC (program counter)
  - SP (stack pointer)
  - General registers
  - Interrupt and exception flags/registers
  - Floating point registers
- The structure that holds this information is hardware dependent? (Why?)

Resource Information

- The list of constraints on how much of any given resource the process can use.
- File information
  - A collection of open files (including things like file pointers)
  - The identity of the current working directory
- Scheduling information
  - How much time the process has used
  - Its priority
  - Statistics about its resource consumption

Address Space

- Mostly bookkeeping
- Do not need to keep copies of memory, just records of where the memory is or how to find it.
    - Virtual memory management (details later)

Other Attributes of a Process

- Identity information
  - A unique id, often called a process id (PID).
- Credentials
  - A way of describing what permissions or capabilities a process has.
  - Example: superuser, administrator
- Signal information
  - Which signals this process catches
  - Which signals this process ignores

Process Data Structures (1)

- Without careful design, process management structures can become unwieldy!
- It is useful to think about the parts of a process in a variety of different ways:
  - Which parts of the process are machine (in)dependent?
  - Which parts need to be per-thread and which are pre-process?
- How do you find information for a particular process?
- How do you map PIDS to/from processes?

Process Data Structures (2)

- Typically you have one data structure to encapsulate an entire process.
    - Traditionally called a process control block (PCB).
    - Called a task struct in Linux.
    - Called a proc struct in *BSD.
- PCB typically contains references to other structures:
    - Address space
    - Threads associated with the process
    - Credentials
- All the PCBs are gathered together into a process table.
- Examples:
    - Linux task structure
    - BSD proc structure