/proc

Ways to Interact with OS

- System Calls
  - Message passing
  - Rigid interface
  - Portable
  - Good documentation
  - More on this in project 2
- Proc File System
  - File communication
  - More flexible interface
  - Less portable
  - Little documentation
  - Used in project 1

Proc File System

- Virtual file system
- Pass information to/from kernel via files
- Typically located at the root of the file system
  - /proc
- Useful for kernel debugging and for accessing data that doesn't necessarily warrant a system call
- Essentially a backdoor into the operating system
  - Can lead to security implications if you're not careful

Proc Examples

- Time since last reboot
  - cat /proc/uptime
- Linux and gcc version info
  - cat /proc/version
- Accessible devices and partitions
  - cat /proc/partitions
- Memory mapped I/O regions
  - cat /proc/iomem

Proc Examples

- Clear file system cache from memory
  - echo “3” | sudo tee /proc/sys/vm/drop_caches
- Max buffer size of pipes
  - cat /proc/sys/fs/pipe-max-size
- Global maximum on number of open files
  - cat /proc/sys/fs/file-max

Proc Examples

- Soft and hard limits impossed on a process
  - cat /proc/<pid>/limits
- Process status
  - cat /proc/<pid>/status

limits

- Command you have to implement
  - limits cmd
- Executes provided command
- Then prints out information about the soft, hard limits imposed on that process
- Only print out
  - Max file size
  - Max open files
  - Max processes
  - Max pending signals

limits Workflow

- Child
  - Execute cmd
  - Exit
- Parent
  - Get pid of child
  - Get data in /proc/<pid>limits
  - Wait for child to finish
  - Print out relevant lines