



Processes and User Accounts

CIS 68C1-01

Lecture 3

Instructor: Mike Cappella

Process Components

■ What is a Process?

- A process is an *instantiation* of a program
- Kernel data structures used to manage RAM, CPU usage, I/O, process status, etc.

■ PID

- Process ID
- Unique, recyclable integer process identifier

■ UID / GID

- User ID / Group ID
- Determines owner/group of process

Process Components

■ EUID / EGUID

- Effective User ID / Effective Group ID
- Determines access permissions

■ Nice Value

- Influences amount of CPU time kernel gives
- Nice processes receive less CPU time

■ Controlling Terminal

- STDIN, STDOUT, STDERR

Process Life Cycle

■ Process Birth

- First, a process clones itself
 - `fork()` system call
- Then the cloned process overlays itself with a new program
 - `exec()` system call
- All processes are created using **fork / exec**
 - Except **init**, which is created by the kernel during bootstrap

Process Life Cycle

■ Process Life

- The cloning process is the *parent* process
- The cloned process is the *child* process
- Parent waits for child process to die
 - Using wait() system call
- Child process runs independently until completion

Process Life Cycle

■ Process Death

□ Completed process

- Exits with an **exit code** indicating reason for exiting
- Is now a **zombie**

□ Parent process receives child's exit code

- The `wait()` system call returns
- Kernel cleans up completed process
 - No longer a zombie

Process Life Cycle

■ Process Death

- What if parent...

- dies before child?

- does not call `wait()`

- The **init** process waits for any child process whose parent does not wait

Signals

- Signals are process-level interrupt requests
 - Small messages delivered to a process by the kernel
 - There are about 30 different signals
- Cause processes to take some action
- A process may install a *signal handler* to handle the signal
- Otherwise, kernel will take some signal-dependent, default action

Signals

- Signals can be
 - Caught - Managed internally by process
 - Blocked - Not received by process until unblocked
 - Ignored - Not delivered to process (discarded)
- Kernel applies default action if signal is not **caught, blocked, or ignored**

Signals

- Each signal has its own default action
 - Terminate process
 - Stop Process
 - Ignored
- Two signals, KILL and STOP, can **never** be caught, blocked, or ignored
 - KILL (signal 9) causes kernel to terminate process
 - STOP causes process to stop running

Signals

- Some signals appear to do the same thing
 - KILL, INT, TERM, QUIT, HUP
- KILL
 - Cannot be blocked
 - Never delivered
 - Process terminated by kernel
- INT
 - Interrupt – sent by terminal driver in response to ^C
 - Process should quit or clean-up as appropriate

Signals

■ TERM

- Request to terminate – process *should* clean up and quit

■ QUIT

- Similar to TERM, but also causes code dump

■ HUP

- Hang-up - sent by terminal driver (e.g. when connection is lost)
- Used by daemons as request for restart

Signals

- Signals are sent using the **kill** command
- Synopsis
 - `kill [-signal] pid`
- TERM signal is default signal
- `kill -9 pid`
 - Forces process *pid* to be killed

Process States

- A process is always in one of four states
 - Runnable
 - The process is ready to run
 - Sleeping
 - The process is waiting for an event to occur
 - Stopped
 - Process cannot run, until receipt of CONT signal
 - Zombie
 - Process is dead, awaiting clean up

Process Niceness

- Process Nice Value (or niceness)
 - Influences time process receives on CPU
 - High value = less CPU time; low value = more
 - Nice values vary across UNIX systems
 - RedHat: -20 to 20
 - Process inherits nice value of parent
 - Nice value set with **nice** / **renice** commands
 - The **nice** command is built into some shells

Process Examination

■ ps

- Utility to examine status of processes
- Two forms of **ps**
 - BSD-based
 - System V

■ top

- Shows process information updated periodically in real-time

Process Runaways

- Processes can cause trouble if they are resource hogs (CPU time, disk space, etc.), or have gone amuck
- Examine process information with **ps**
 - CPU usage, size, nice value
- Learn more about process before you kill it
- Consider stopping it with STOP signal

User Accounts

- User account information lives in the password file `/etc/passwd`
- Each line in `/etc/passwd`
 - Represents a single user account
 - Has 7 colon-separated fields

User Accounts

- Each line in `/etc/passwd` contains
 - Login name
 - Encrypted password
 - UID
 - GID
 - GECOS information
 - Home directory
 - Login shell

User Accounts

■ Login Name

- Used to log into system
- Should limit to 8, lower-case, alpha-numeric characters
- Should create standardized naming scheme
- Should be unique across systems
- Use same login name for same person on additional systems

User Accounts

- Encrypted password
 - Set to * to disable account or when creating account
 - Set to **x** to use /etc/shadow file
 - /etc/shadow file is used to store and keep encrypted password from view

User Accounts

■ UID

- Numeric ID representing a User
- Newer systems have 32-bit UID value
 - May need to limit values to $< 32,767$ or $65,535$ for compatibility with older systems
- UID 0 is always **root** account
- Keep UIDs unique and consistent across all systems for NFS

User Accounts

■ Group ID

- Numeric ID, similar to UID
- Default group for user at login
- Groups defined in `/etc/group`
- User can be in several groups
- Keep GIDs consistent across systems

User Accounts

■ GECOS Field

- Not well-defined
- Can be used to record personal information
- Typically used now for user's full name
- The finger utility uses this information
- The **chfn** command allows user to change data
 - Often **chfn** is disabled by sys admins

User Accounts

■ Home Directory

- Current working directory on login
- UID/GID should be set to user's UID/GID
- Home directories are often mounted over NFS
 - Network problems may cause unavailability

User Accounts

■ Login Shell

- The shell the user will use upon login
- The **chsh** command lets users change shell to valid shell
 - /etc/shells is the list of valid shells that chsh allows
 - Sys admins often prevent users from changing shells

User Accounts

■ /etc/shadow

- Contains one line / user
- Each line is 9 colon-separated fields
- The login name and password fields cannot be empty
 - Login name is same as that in /etc/passwd
 - Password is created by running **passwd** command

User Accounts

- /etc/group
 - Contains names and members of groups
 - Each line is 4 colon-separated fields
 - Group name
 - Encrypted password
 - GID number
 - List of members (comma separated)

User Accounts

■ Adding New Users

- Add line for new user into `/etc/passwd` and `/etc/shadow`, filling in appropriate fields
- Set initial password using **passwd** command
- Create users home directory
- Copy default startup files
- Add user to `/etc/group` file
- Change permissions and ownership of user's files and home directory
- Verify login by testing

User Accounts

- Disabling User Accounts
 - Can remove account
 - Can change shell to program that outputs Account Disabled message and then exits
 - Program should not be in /etc/shells