# Processes and Daemons

## Advanced Embedded Linux Development

with **Dan Walkes**

University of Colorado **Boulder**

**Learning objectives:**

Process users/groups and permissions

The ps utility.

Introduction to Daemons.

# Users/Groups and Processes

- There are Multiple User/Group IDs associated with a process
  - Track who originally ran the process (real user and group ids)
  - Track who the process is currently running as
    - effective user ID
    - typically the most useful.

# Users/Groups and Processes

- Effective user ID can be changed with setuid or setgid()
- Real user (who started the process) can be obtained with getuid(), getgid().
- Effective user and group with geteuid(),getegid()

# User Group Selection

- What user/group should I use for my process?
  - root account - ensured access to whatever I need
  - specific other account
- Doctrine of least-privileged rights
  - Process should execute with minimum level of rights possible.

# Doctrine of least-privileged rights

- Process should execute with minimum level of rights possible.
- Why?
  - Less susceptible to exploit
  - Less susceptible to errors
    - rm -rf ${MISTYPEDD_VAR}/

# Process Groups and Sessions

- Process Group is a collection of processes
- Session is a collection of Process Groups
  - Associated with a controlling terminal
    - Also known as tty (TeleTypewriter)
    - tty is a device for terminal I/O (keyboard interaction, login, command input/output)
  - A session is created for the login shell on a tty.

Linux System Programming Chapter 5
https://askubuntu.com/questions/481906/what-does-tty-stand-for

# Process Groups and Sessions

- Single foreground process group in a session
  - This is the one interrupted with Ctrl->C
- 0 or more background process groups.
  - Adding “&” after a shell command puts the command in a background process group

# ps Utility

- ps shows status about processes

```
NAME
       ps - report a snapshot of the current processes.

SYNOPSIS
       ps [options]

EXAMPLES
       To see every process on the system using standard syntax:
          ps -e
          ps -ef
          ps -eF
          ps -ely

       To see every process on the system using BSD syntax:
          ps ax
          ps axu

       To print a process tree:
          ps -ejH
          ps axjf

       To get info about threads:
          ps -eLf
          ps axms
```

http://man7.org/linux/man-pages/man1/ps.1.html

# ps Utility

```
ecen5013@ecen5013-VirtualBox:~$ ps aux | less
```

```
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0 225524  7388 ?        Ss   Sep09   0:03 /lib/systemd/systemd --system --deserialize 41
root           2  0.0  0.0      0     0 ?        S    Sep09   0:00 [kthreadd]
root           3  0.0  0.0      0     0 ?        I<   Sep09   0:00 [rcu_gp]
root           4  0.0  0.0      0     0 ?        I<   Sep09   0:00 [rcu_par_gp]
root           6  0.0  0.0      0     0 ?        I<   Sep09   0:00 [kworker/0:0H-kb]
root           8  0.0  0.0      0     0 ?        I<   Sep09   0:00 [mm_percpu_wq]
root           9  0.0  0.0      0     0 ?        S    Sep09   0:07 [ksoftirqd/0]
root          10  0.0  0.0      0     0 ?        I    Sep09   0:18 [rcu_sched]
root          11  0.0  0.0      0     0 ?        S    Sep09   0:03 [migration/0]
root          12  0.0  0.0      0     0 ?        S    Sep09   0:00 [idle_inject/0]
```

```
ecen5013   1803  0.0  0.0  76912  3644 ?        Ss   Sep09   0:00 /lib/systemd/systemd --user
ecen5013   1804  0.0  0.0 261880  2896 ?        S    Sep09   0:00 (sd-pam)
ecen5013   1925  0.0  0.0 110072  2792 ?        S    Sep09   0:09 sshd: ecen5013@pts/0
ecen5013   1926  0.0  0.0  29912  5040 pts/0    Ss   Sep09   0:00 -bash
ecen5013   1989  0.0  0.0  39380  3000 ?        Ss   Sep09   0:09 SCREEN -S build
ecen5013   1990  0.0  0.0  29936  4924 pts/1    Ss   Sep09   0:00 /bin/bash
```

```
ecen5013 32474  0.0  0.0  44472  3424 pts/1    R+   07:59   0:00 ps aux
ecen5013 32475  0.0  0.0  16956  1088 pts/1    S+   07:59   0:00 less
```

- Root processes
- User processes
- The ps command

# Daemons

- Originally pronounced “dee-men”, also commonly pronounced “day-men”
- A process which runs in the background, does not connect to a controlling terminal.
  - Typically started at boot time
  - run as root or a special user
  - Often end with d in name (sshd, crond)
- Often runs as a child of init

# How to create a Daemon

- fork()
  - Creates a new child process which will become the daemon
- exit() in parent
  - allows daemon's grandparent (init) to continue

# How to create a Daemon

- setsid()
  - Creates a new session (ensure no controlling tty)
- chdir(“/”)
  - Change working dir
- Close file descriptors
- Redirect stdin, stdout, stderr to /dev/null

# Redirect of std* in Daemon

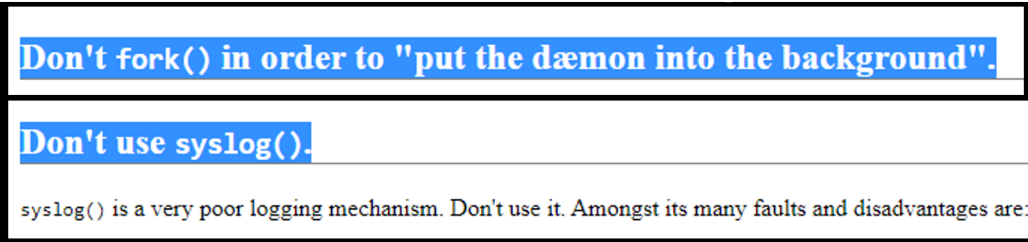
- Why redirect stdin, stdout, and stderr to /dev/null? Where would it go if we didn't redirect?
  - The terminal if started from the terminal
- What about redirect to a file instead?
  - What if the file is on a filesystem which is trying to be unmounted?

# Redirect of std* in Daemon

- How do I give feedback to the user about status/errors within my daemon?
  - Use logging framework like syslog

# Different Daemon Options

- Like anything else in Linux, not everyone agrees with the author (or with each other)
  - https://jdebp.eu/FGA/unix-daemon-design-mistakes-to-avoid.html

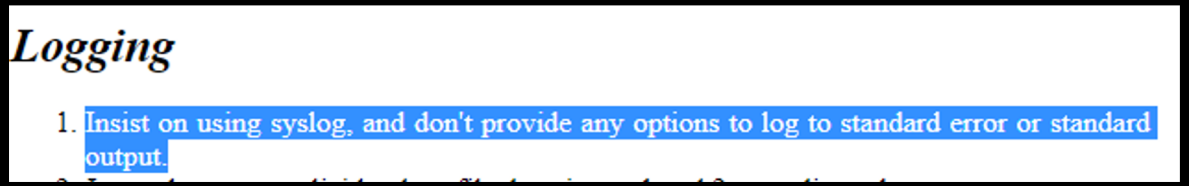


**Don't `fork()` in order to "put the dæmon into the background".**

**Don't use `syslog()`.**

`syslog()` is a very poor logging mechanism. Don't use it. Amongst its many faults and disadvantages are:

  - http://cloud9.hedgee.com./scribbles/daemon#logging



*Logging*

1. Insist on using syslog, and don't provide any options to log to standard error or standard output.