CptS 360 (System Programming)
Unit 12: Process Relations

Bob Lewis

School of Engineering and Applied Sciences
Washington State University

Spring, 2022

Motivation

- Processes are fundamental components of operating systems.
- “How Does a Shell Work?”

References

- Stevens & Rago Ch. 9
- *man* pages

4.3BSD (and similar UNIX) Terminal Logins

Remember: *systemd(1)* (aka *init*, PID 1, see below) forks itself and *exec*()*s *getty(1)* on all tty lines specified in */etc/init/tty*.conf*.

getty(1)

- opens terminal device
- reads user name
- sets initial environment
- *exec*()*s *login(1)*

login(1)

- gets and verifies password
- `cd`'s to home directory
- *exec*()*s `$SHELL` as if

```
execl($SHELL, "-" + $(basename $SHELL), NULL)
```

4.3BSD (and similar UNIX) Terminal Logouts

When login shell exits, *systemd(1)*

- (ultimately) gets `SIGCHLD`
- starts *getty(1)* over again on tty line

What About *Graphical* Logins?

- now there's a "display manager" (named `*dm`, by convention) involved
- *take a look at pstree(1)*
- Q: Why are the "gnome-*" utils children of "systemd"?

systemd: A New, Improved *init(1)*

Improvements:

- moves what used to be in `/etc/inittab` (as described in S & R) to `/etc/init/` (q.v.)
- clean, straightforward, and efficient design
- simpler boot process
- concurrent and parallel processing at boot
- better API
- simple unit syntax
- ability to remove optional components
- low memory footprint
- improved technique to express dependencies
- initialization instruction written in config file and not in shell script
- make use of Unix Domain Socket
- job scheduling using systemd calendar timers
- event logging with journald
- choice of logging System events with systemd as well as syslog
- logs are stored in binary file
- systemd state can be preserved to be called later in future
- track process using kernel's cgroup and not PID
- users login managed by systemd-logind
- better integration with Gnome for interoperability

Network Logins

What happens when you log in to a machine over a network?

- *systemd(1)* starts up *inetd(8)*
- TCP connection causes `inetd` to
  - fork
  - exec *sshd(8)* (or, in olden days, *telnetd(8)*)
  - `sshd` (or `telnetd`) starts *login(1)*, which then behaves as above console but uses "pseudo-tty"s (explain)

Process Groups

Processes are organized into groups, mainly for signal delivery purposes.

- calls:
  - *setpgrp(2)*
  - *setpgid(2)*
    - This creates a new process group
    - takes `pid_t pid`, `pid_t pgid`
  - *getpgid(2)*
  - *getpgrp(2)*
    - takes `pid_t pid`, `pid_t pgid`
    - sets the process GID of pid to pgid.

Job Control

In job control shells,

- After *fork()*, both parent and child call *setpgrp(2)*.
- One call is redundant, but this avoids a race condition by guaranteeing that the child is in a different process group.

Sessions

- A session is a collection of process groups.
- Process groups determine signal delivery, but sessions determine “controlling tty” (see below).
- *setsid(2)*
  - If the calling process is not already a session owner, a new session is created with no controlling tty.
  - It’s an error if it is already a session owner.
  - It also belongs to a newly-created process group.
  - A preceding *fork()* with parental *exit()* guarantees that the call will not fail.

Controlling Terminal

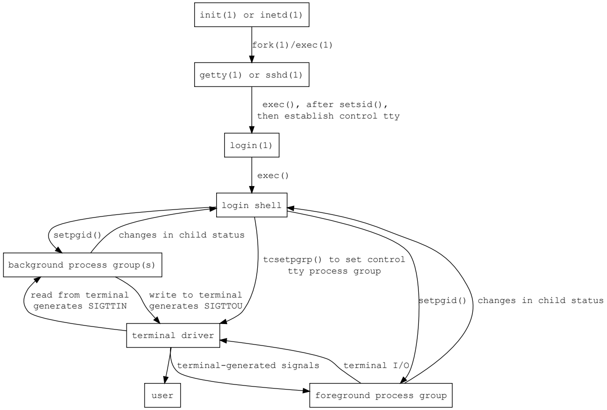
- A process can have at most one of these.
- Session gets controlling terminal (`tty` or `pty`).
- Within a session, there's a foreground process group and 0 or more background process groups.
- Keyboard interrupts go to the foreground process group only.
- Any process can get to its controlling tty (if it has one) by opening `/dev/tty`.
- Controlling tty set (a la BSD) via

  ```
  ioctl(fdtty, TIOCSCTTY, NULL)
  ```

tcgetpgrp(2) and *tcsetpgrp(2)*

sets controlling process group

Job Control


init(1) or inetd(1)
fork(1)/exec(1)
getty(1) or sshd(1)
exec(), after setsid(), then establish control tty
login(1)
exec()
login shell
setpgid()
changes in child status
background process group(s)
tcsetpgrp() to set control tty process group
setpgid()
changes in child status
read from terminal generates SIGTTIN
write to terminal generates SIGTTOU
terminal driver
terminal-generated signals
terminal I/O
user
foreground process group

(after S & R Figure 9.9)

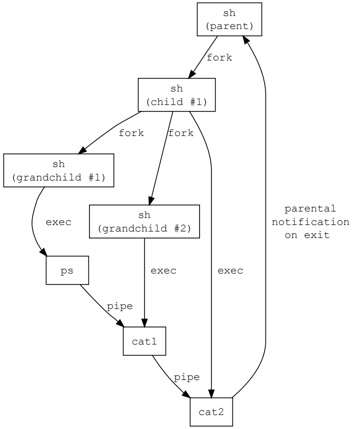
Shell Execution of Programs

result of

`ps | cat1 | cat2`

after S & R Figure 9.9

- How are the pipes set up?



Orphaned Process Groups

- processes can become orphans
- so can whole groups
- demo Stevens & Rago Program 9.11