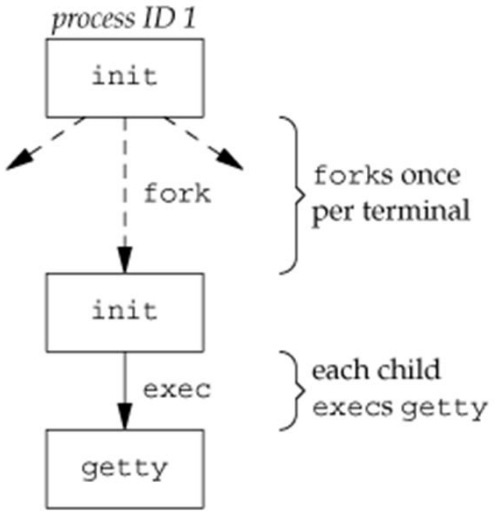
Process relationships

Terminal Logins

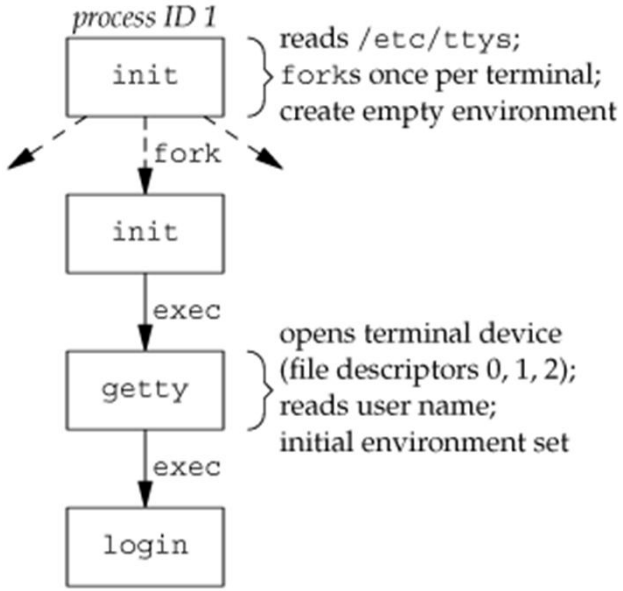
1. init forks once per terminal.
2. each child of init execs getty.



Processes invoked by init to allow terminal logins.

Terminal Logins

3. getty opens for terminals and then waits for us to enter our user name.
4. When we enter our user name, getty execs login.



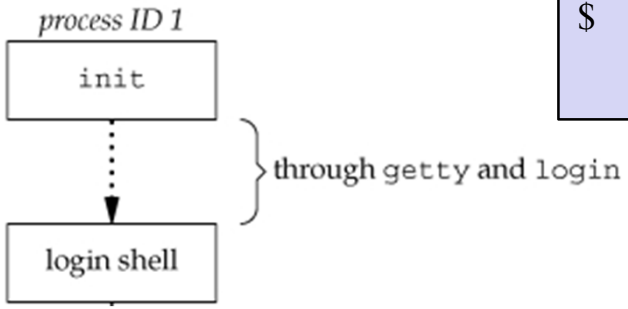
login as: obama
obama's password:

State of processes after login has been invoked.

Terminal Logins

5. login reads password and authenticates.
6. If we log in correctly, login changes to our home directory, changes ownership of our terminal device, and initializes environment variables.
7. login execs our login shell, execl("/bin/bash", "-bash", 0);

```
login as: obama
obama's password: ******
Last login: Mon May 18 22:12:20 …
$
```



Processes after everything is set.

Process groups

- Process group
  - A collection of one or more processes.
  - Usually associated with the same job.
  - Can receive signals from the same terminal.
  - **$ proc1 | proc2** **$ ls | more**

Process groups

- Process group ID
  - Each process group has a unique PGID.
  - Each process group can have the process group leader, whose PID equals its PGID.
  - The process group exists, as long as there is at least one process in the group, regardless whether the group leader terminates or not.

```
$ ps -o pid,ppid,pgid,comm | cat
  PID    PPID    PGID    COMMAND
27463   27462   27463    bash
27554   27463   27554    ps
27555   27463   27554    cat
$
```

Process groups

```
#include <unistd.h>

pid_t getpgrp(void);
                                    Returns: process group ID of calling process
```

- Returns the process group ID of the calling process.

Process groups

```
#include <unistd.h>

pid_t getpgid(pid_t pid);
                                        Returns: process group ID if OK, -1 on error
```

- Return the process group ID of the process with *pid*.
  - If pid is 0, return the process group ID of the calling process.
    - getpgid(0); is equivalent to getpgrp();

Process groups

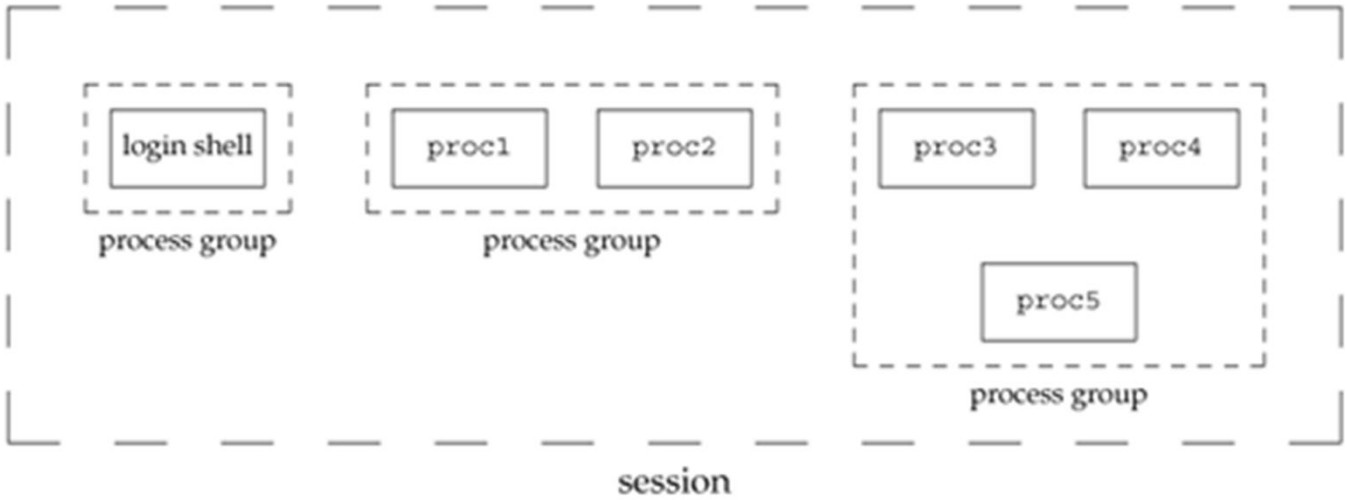
```
#include <unistd.h>

int setpgid(pid_t pid, pid_t pgid);
                                        Returns: 0 if OK, -1 on error
```

- Sets the process group ID of process with *pid* to *pgid*.
  - If pid is 0, the process ID of the current process is used.
  - If pgid is 0, the process ID of the process with *pid* is used as the process group ID.

Sessions

- Session
  - A collection of one or more process groups.
  - $ proc1 | proc2 &
  - $ proc3 | proc4 | proc5



Sessions

- Session example

```
$ ps -o pid,ppid,pgid,session,comm | cat &
[1] 27585
$   PID     PPID     PGID     SESS     COMMAND
27463     27462     27463     27463     bash
27584     27463     27584     27463     ps
27585     27463     27584     27463     cat

[1]+  Done                    ps -o pid,ppid,pgid,session,comm | cat
$ ps -o pid,ppid,pgid,session,comm | cat | cat
PID     PPID     PGID     SESS     COMMAND
27463     27462     27463     27463     bash
27586     27463     27586     27463     ps
27587     27463     27586     27463     cat
27588     27463     27586     27463     cat
[tskim@oslab ~]$
```

Sessions

```
#include <unistd.h>

pid_t setsid(void);
                                    Returns: process group ID if OK, -1 on error
```

- create a new session.
  - The calling process becomes the leader of the new session.
  - The calling process becomes the process group leader of the new process group.

Sessions

```
#include <unistd.h>

pid_t getsid(pid_t pid);
                        Returns: session leader's process group ID if OK, 1 on error
```

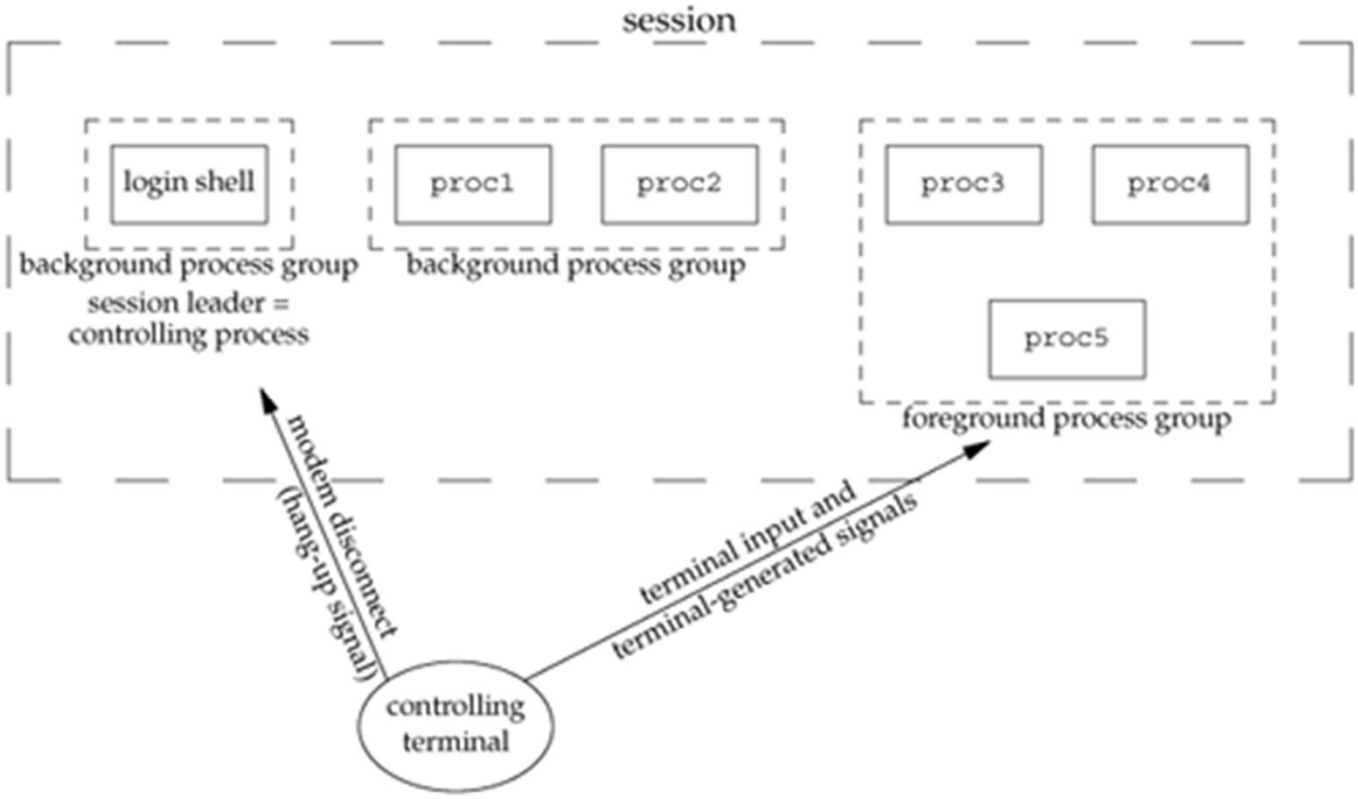
- returns the session ID of the process with *pid*.
  - getsid(0) returns the session ID of the calling process.

Controlling terminal

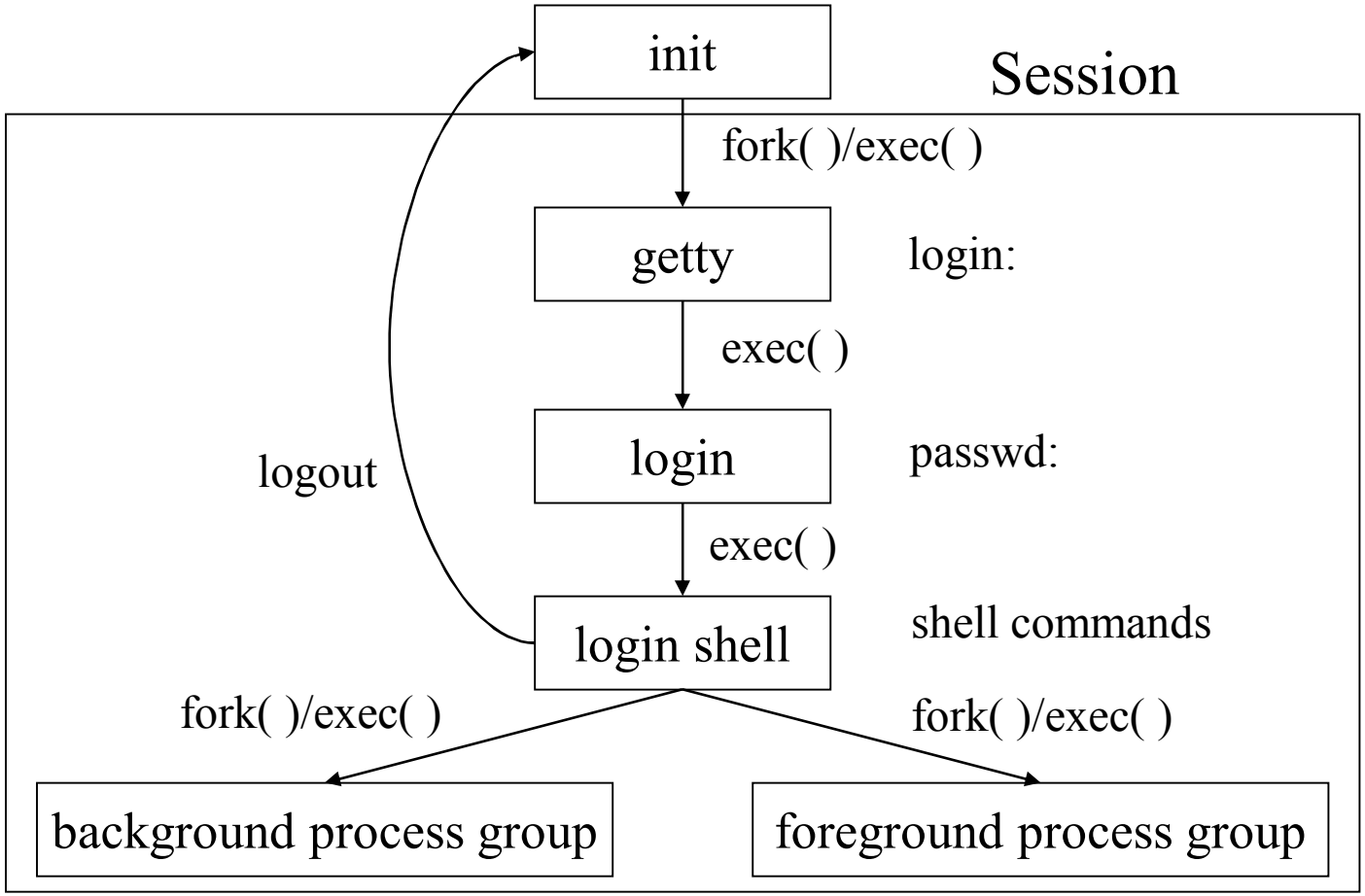
- controlling terminal
  - A session can have a single controlling terminal.
    - Controlling terminal is usually the terminal device.
    - /dev/tty
  - A session may have a single foreground process group and one or more background process groups.
  - The session leader that established the connection to the controlling terminal is called controlling process.
  - interrupt or quit signal are sent to all processes in the foreground process group.
  - hang-up signal is sent to the controlling process.

Controlling terminal

- Process groups and sessions showing controlling terminal



Login and session summary



FreeBSD implementation

- Sessions and process groups