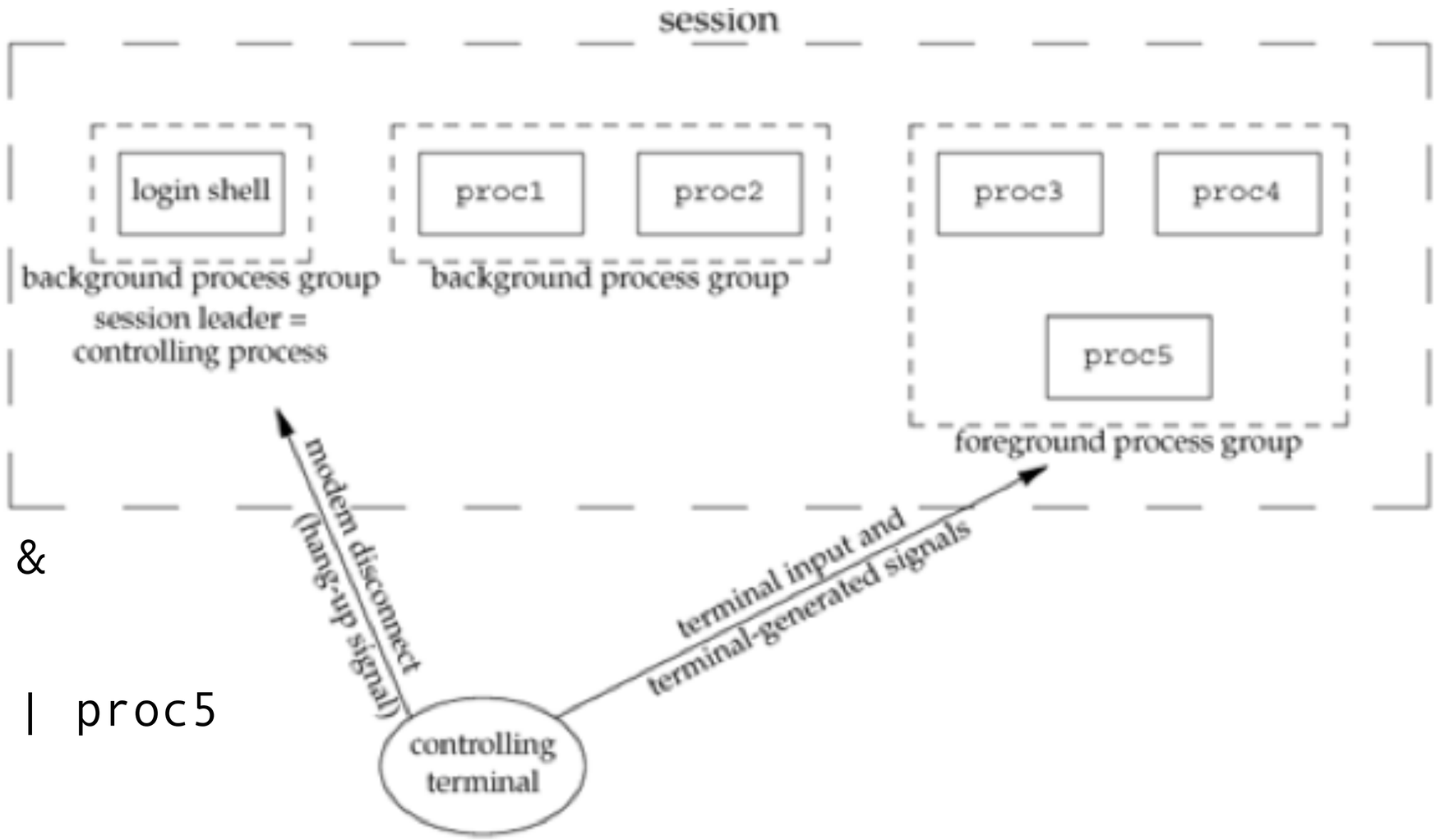
Advanced Programming in the UNIX Environment

Week 07, Segment 3:
Job Control

Department of Computer Science
Stevens Institute of Technology

Jan Schaumann
jschauma@stevens.edu
https://stevens.netmeister.org/631/

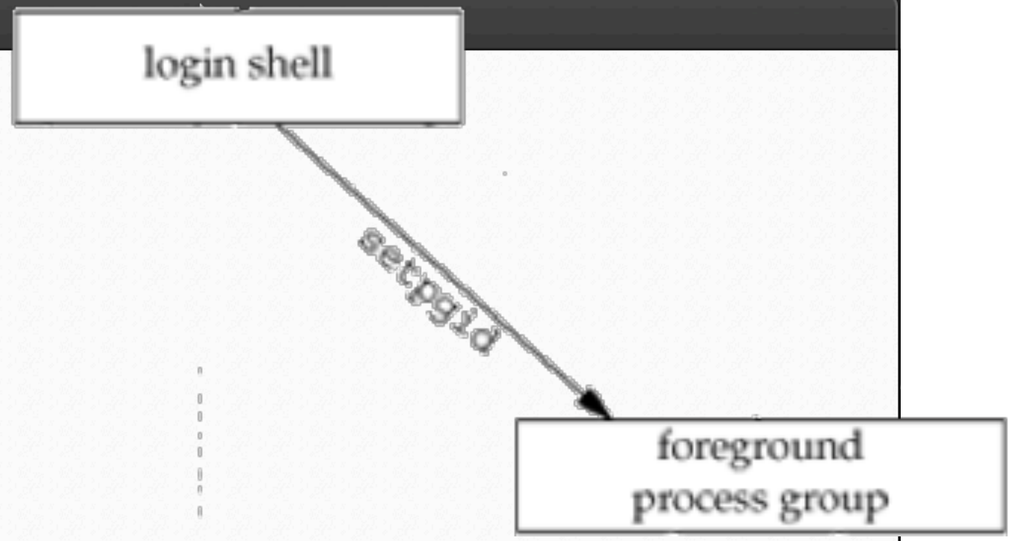
Process Groups




```
$ proc1 | proc2 &
[1] 10306
$ proc3 | proc4 | proc5
```

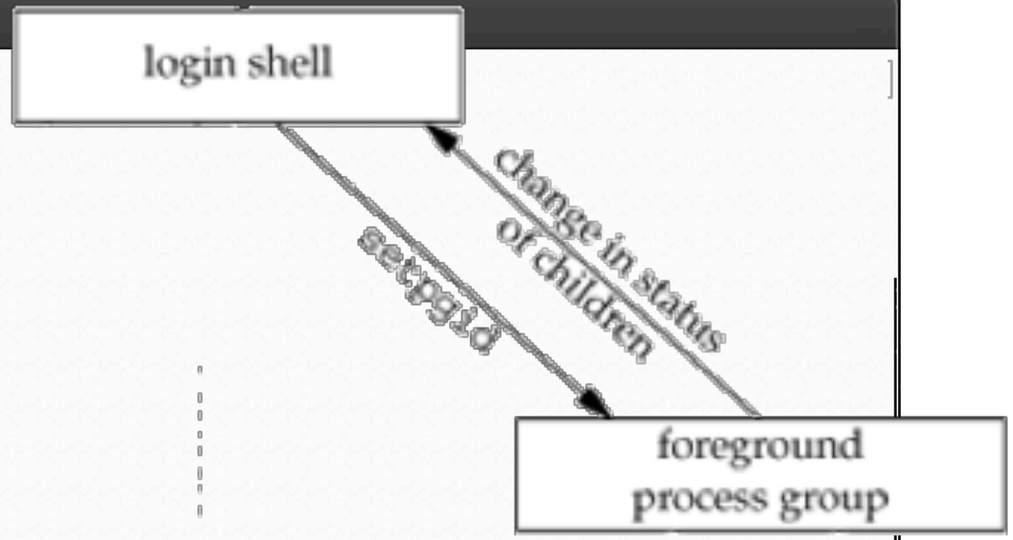
Terminal — 80×24

apue$



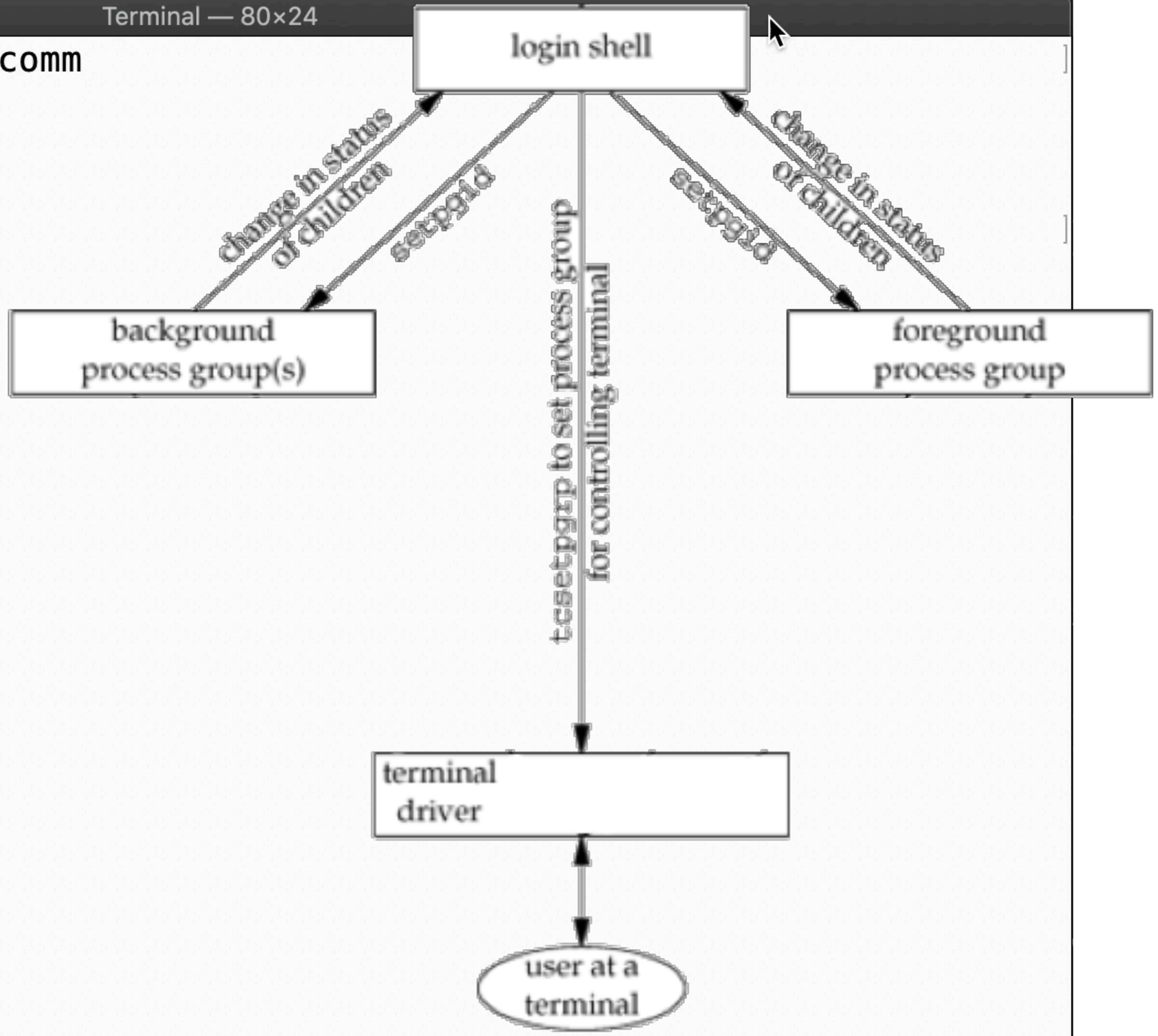
Terminal — 80×24

```
apue$ ps -o pid,ppid,pgid,sid,comm
PID PPID PGID SID COMMAND
 41  711   41  41 -sh
753   41  753  41 ps
apue$
```



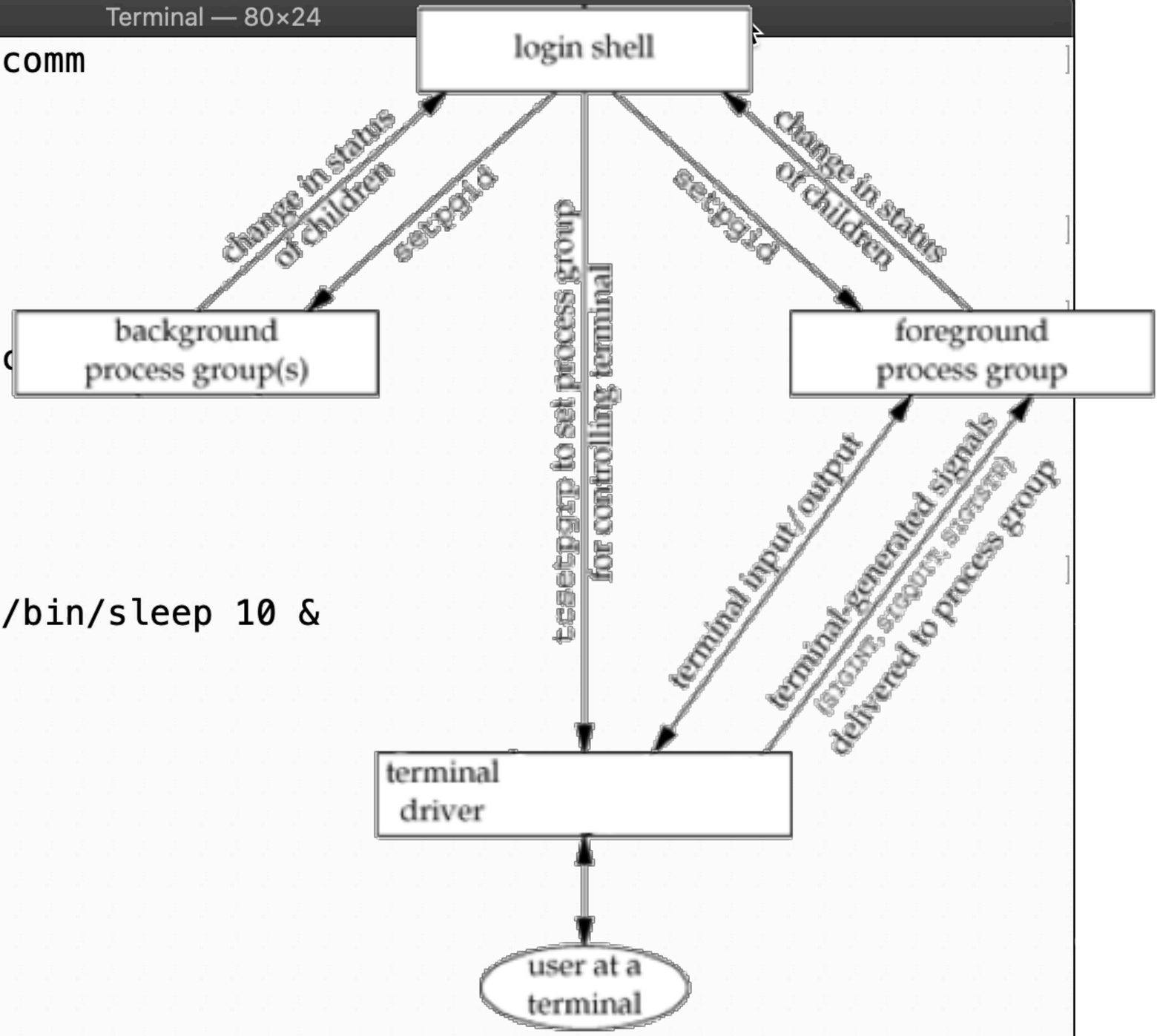
Terminal — 80×24

```
apue$ ps -o pid,ppid,pgid,sid,comm
PID PPID PGID SID COMMAND
 41  711   41  41 -sh
753   41  753  41 ps
apue$ echo $?
0
apue$
```

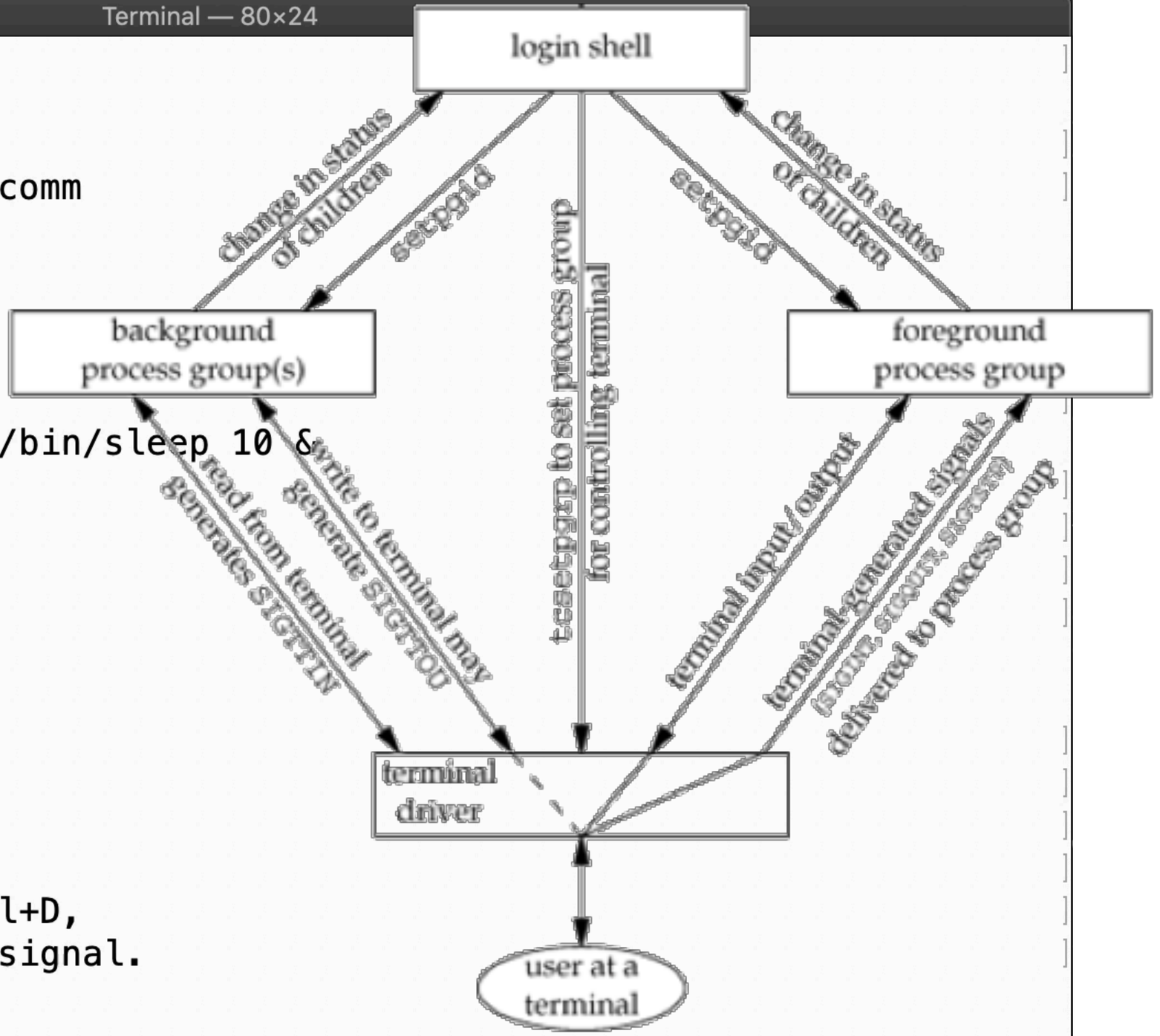


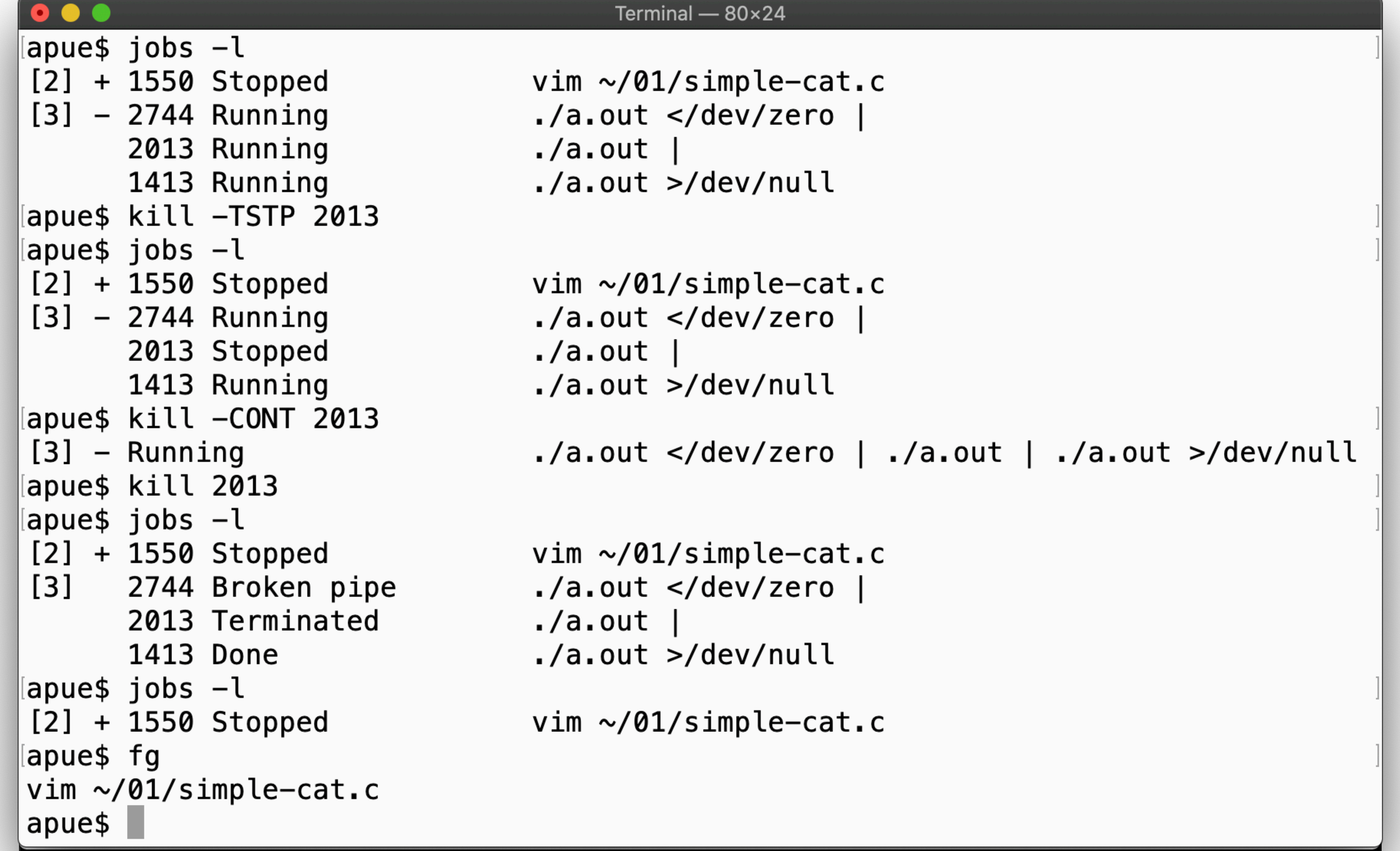
Terminal — 80×24

```
apue$ ps -o pid,ppid,pgid,sid,comm
PID PPID PGID SID COMMAND
 41  711   41  41 -sh
753   41  753  41 ps
apue$ echo $?
0
apue$ /bin/sleep 10 &
apue$ ps -o pid,ppid,pgid,sid,c
PID PPID PGID SID COMMAND
 41  711   41  41 -sh
880   41  880  41 ps
894   41  894  41 /bin/sleep
apue$
[2]   Done                    /bin/sleep 10 &
apue$
```



```
apue$ echo $?
0
apue$ /bin/sleep 10 &
apue$ ps -o pid,ppid,pgid,sid,comm
PID PPID PGID SID COMMAND
 41  711   41  41 -sh
880   41  880  41 ps
894   41  894  41 /bin/sleep
apue$
[2]   Done                    /bin/sleep 10 &
apue$ cat >file
Input from the terminal.
Output to the terminal.
apue$ cat file
Input from the terminal.
Output to the terminal.
apue$ cat >/dev/null
Same here.
cat(1) calls read(2), which
now happily blocks forever.
We can either send EOF via Ctrl+D,
or we could send an interrupt signal.
^C
apue$
```




Terminal — 80×24
apue$ jobs -l
[2] + 1550 Stopped vim ~/01/simple-cat.c
[3] - 2744 Running ./a.out </dev/zero |
2013 Running ./a.out |
1413 Running ./a.out >/dev/null
apue$ kill -TSTP 2013
apue$ jobs -l
[2] + 1550 Stopped vim ~/01/simple-cat.c
[3] - 2744 Running ./a.out </dev/zero |
2013 Stopped ./a.out |
1413 Running ./a.out >/dev/null
apue$ kill -CONT 2013
[3] - Running ./a.out </dev/zero | ./a.out | ./a.out >/dev/null
apue$ kill 2013
apue$ jobs -l
[2] + 1550 Stopped vim ~/01/simple-cat.c
[3] 2744 Broken pipe ./a.out </dev/zero |
2013 Terminated ./a.out |
1413 Done ./a.out >/dev/null
apue$ jobs -l
[2] + 1550 Stopped vim ~/01/simple-cat.c
apue$ fg
vim ~/01/simple-cat.c
apue$

Job Control

- both background and foreground process groups may report a change in status to the login shell
- the foreground process group can perform I/O on the controlling terminal
- the controlling terminal can generate signals via keyboard interrupts to send to the foreground process group
- the background process group may be able to write to the controlling terminal
- the background process group may generate a signal to send to the controlling terminal if it needs to perform I/O
- the shell may move process groups into the foreground or background, suspend or continue them
- we can send any signal to any process via `kill(1)`