**Chapter 9.9: Shell Execution of Programs**

**ps Command**

Bourne shell running on Solaris, command:

ps -o pid,ppid,pgid,sid,comm

Output:

PID PPID PGID SID COMMAND

949 947 949 949 sh

1774 949 949 949 ps

Parent of ps is the shell.

Both shell and ps command are in the same session and foreground process group (949).

949 is the foreground process group because that is what you get when you execute a command with a shell that doesn’t support job control

Command running in the background

ps -o pid,ppid,pgid,sid,comm &

Output:

PID PPID PGID SID COMMAND

949 947 949 949 sh

1812 949 949 949 ps

Only the PID changes; background job is not placed into its own process group and the controlling terminal isn’t taken away from the background job because this shell doesn’t know about job control

Command running with pipeline

ps -o pid,ppid,pgid,sid,comm | cat1

Output:

PID PPID PGID SID COMMAND

949 947 949 949 sh

1823 949 949 949 cat1

1824 1823 949 949 ps

Last process in the pipeline is a child of the shell, and the first process in the pipeine is a child of the last process.

Shell forks a copy of itself, the copy then forks to make each of the previous processes in the pipeline

Command running with pipeline in the background

ps -o pid,ppid,pgid,sid,comm | cat1 &

Only the process IDs will change, since the shell doesn’t handle job control. The process group ID of the background processes and process group ID remains 949.

If a background process tries to read from its controlling terminal:

cat > temp.foo &

With job control:

The background job is placed into a background process group which causes SIGTTIN to generate when the job tries to read from its controlling terminal.

Without job control:

Shell automatically redirects the standard input of the background job into /dev/null/ if the process doesn’t redirect standard input itself. Cat process will terminate because it will immediate read an EOF generated by /dev/null/.

If background process specifically opens /dev/tty/ and reads from controlling terminal:

crypt < salaries | lpr &

crypt opens up /dev/tty/, reads from the device and resets the terminal characteristics

crypt generates the password prompt, but input is read by shell that tries to execute the command of that password name. The next line is then read as the password, so the file isn’t encrypted correctly, and junk is sent to the printer.

Executing 3 process in a pipeline

ps -o pid,ppid,pgid,sid,comm | cat1 | cat2

Output:

PID PPID PGID SID COMMAND

949 947 949 949 Sh

1988 949 949 949 cat2

1989 1988 949 949 ps

1990 1988 949 949 cat1

Last process of the pipeline is the child of the shell, and all previous processes are children of the last process.

Executing command in background with job control

ps -o pid,ppid,pgid,sid,tpgid,comm &

Output:

PID PPID PGID SID TPGID COMMAND

2837 2818 2837 2837 5796 bash

5796 2837 5796 2837 5796 ps highlighted = foreground process group

Places foreground job (ps) into its own process group (5796), ps command becomes the process group leader and the only process in this process group. Login shell is background process while the ps command executes.

2837 and 5796 are members of the same session

Executing two processes in a pipeline with job control

ps -o pid,ppid,pgid,sid,tpgid,comm | cat1

output:

PID PPID PGID SID TPGID COMMAND

2837 2818 2837 2837 5799 bash

5799 2837 5799 2837 5799 ps

5800 2837 5799 2837 5799 cat1

Both processes, ps and cat1, are placed into a foreground process group (5799)

Executing two processes in a pipeline in the background with job control

ps -o pid,ppid,pgid,sid,tpgid,comm | cat1 &

output:

PID PPID PGID SID TPGID COMMAND

2837 2818 2837 2837 2837 bash

5801 2837 5801 2837 2837 ps

5802 2837 5801 2837 2837 cat1

Ps and cat1 are placed in the same background process group (5801)

The order in which a shell creates processes can differ depending on the particular shell in use.