JOB CONTROL IN UNIX

- To allow us to start multiple jobs from a single terminal and control which jobs can access the terminal and which jobs are to be run in the background.
- It requires 3 forms of support:
  - A shell that supports job control.
  - The terminal driver in the kernel must support job control.
  - Support for certain job-control signals
  - A job is just a collection of processes, often a pipeline of processes.
  - When we start a background job, the shell assigns it a job identifier and prints one or more process IDs.
  - $ make all > Make.out &

    [1] 1475

    $ pr *.c | lpr &

    [2] 1490

    $ just press RETURN

    [2] + Done pr *.c | lpr &

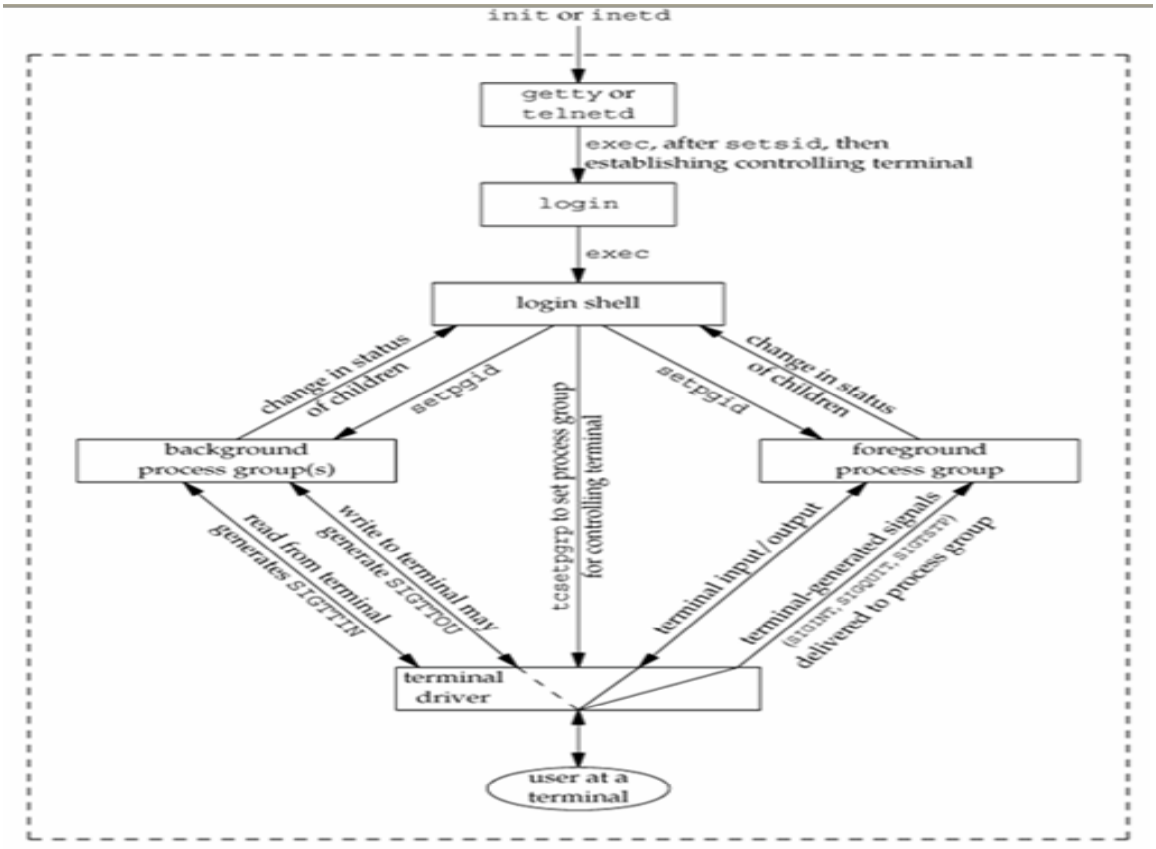
    [1] + Done make all > Make.out &

- The reason why we have to press RETURN is to have the shell print its prompt. The shell doesn't print the changed status of background jobs at any random time -- only right before it prints its prompt, to let us enter a new command line.
- Entering the suspend key (Ctrl + Z) causes the terminal driver to send the SIGTSTP signal to all processes in the foreground process group.

  The terminal driver really looks for 3 special characters, which generate signals to the foreground process group:

- The interrupt character generates SIGINT

- The quit character generates SIGQUIT
- The suspend character generates SIGTSTP



PROGRAM

$cat temp.foo & start in background, but It'll read from standard input

[1] 1681

$ we press RETURN

[1] + Stopped (tty input) cat > temp.foo &

$ fg %1 bring job number 1 to foreground

cat > temp.foo the shell tells us which job is now in the foreground

hello, world enter one line

^D type our end-of-file

$ cat temp.foo check that the one line put into the file

hello, world

- What happens if a background job outputs to the controlling terminal?
- This option we can allow or disallow. Normally we use the stty(1) command to change this option.

```
 $ cat temp.foo &                    execute in background
[1]      1719
$ hello, world                       the output from the background
                                     appears after the prompt we press return
[1] + Done                           cat temp.foo &
$ stty tostop                        disable ability of background jobs to
                                     output to controlling terminal
[1]              1721
$                                    we press return and find the job is stopped
[1] + Stopped(tty output)            cat temp.foo &
```

Shell Execution Of Programs

- Bourne shell doesn't support job control
- ps –xj gives the following output

```
PPID   PID   PGID   SID   TPGID  COMMAND
   1   163   163    163   163      -sh
 163   168   163    163   163      ps
```

- Both the shell and the ps command are in the same session and foreground process group(163). The parent of the ps command is the shell.
- A process doesn't have a terminal process control group.
- A process belongs to a process group, and the process group belongs to a session. The session may or may not have a controlling terminal.
- The foreground process group ID is an attribute of the terminal, not the process.
- If ps finds that the session does not have a controlling terminal, it prints -1.

If we execute the command in the background,

```
Ps –xj &
```

The only value that changes is the process ID.

ps –xj | cat1

PPID	PID	PGID	SID	TPGID	COMMAND
1	163	163	163	163	-sh
163	200	163	163	163	cat1
200	201	163	163	163	ps

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

- If we execute the pipeline in the background

  ps –xj | cat1 &

- Only the process IDs change.
- Since the shell doesn't handle job control, the process group ID of the background processes remains 163, as does the terminal process group ID.

Source : http://elearningatria.files.wordpress.com/2013/10/cse-iv-unix-and-shell-programming-10cs44-notes.pdf