The Unix Shell

Job Control
shell
shell

```
$ wc -l *.pdb | sort | head -1
```
$ wc -l *.pdb | sort | head -1
shell

$ wc -l *.pdb | sort | head -1

Control programs while they run
shell

$ wc -l *.pdb | sort | head -1

Job Control

Introduction

Control programs while they run

processes
A *process* is a running program
A *process* is a running program

Some are yours
A *process* is a running program

Some are yours

Most belong to the operating system (or other users)
A *process* is a running program

Some are yours

Most belong to the operating system (or other users)

Use `ps` to get a list
A *process* is a running program
Some are yours
Most belong to the operating system (or other users)
Use `ps` to get a list

```
$ ps
PID  PPID  PGID  TTY  UID   STIME   COMMAND
2152  1    2152  con  1000  13:19:07 /usr/bin/bash
2276  2152  2276  con  1000  14:53:48 /usr/bin/ps
$  
```
A process is a running program
Some are yours
Most belong to the operating system (or other users)
Use `ps` to get a list

```
$ ps
  PID   PPID   PGID   TTY   UID   STIME   COMMAND
2152    1  2152 con  1000  13:19:07 /usr/bin/bash
2276  2152  2276 con  1000  14:53:48 /usr/bin/ps
$ 
```

Process ID (unique at any moment)
A *process* is a running program

Some are yours

Most belong to the operating system (or other users)

Use `ps` to get a list

```bash
$ ps

<table>
<thead>
<tr>
<th>PID</th>
<th>PPID</th>
<th>PGID</th>
<th>TTY</th>
<th>UID</th>
<th>STIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>2152</td>
<td>1</td>
<td>2152</td>
<td>con</td>
<td>1000</td>
<td>13:19:07</td>
<td>/usr/bin/bash</td>
</tr>
<tr>
<td>2276</td>
<td>2152</td>
<td>2276</td>
<td>con</td>
<td>1000</td>
<td>14:53:48</td>
<td>/usr/bin/ps</td>
</tr>
</tbody>
</table>

$ 
```

Parent process ID
A process is a running program
Some are yours
Most belong to the operating system (or other users)
Use **ps** to get a list

```bash
$ ps
PID   PPID   PGID   TTY   UID     STIME   COMMAND
2152   1   2152   con   1000  13:19:07   /usr/bin/bash
2276  2152   2276   con   1000  14:53:48   /usr/bin/ps
$ 
```

Parent process ID
What process created this one?
A *process* is a running program

Some are yours

Most belong to the operating system (or other users)

Use `ps` to get a list

```
$ ps

<table>
<thead>
<tr>
<th>PID</th>
<th>PPID</th>
<th>PGID</th>
<th>TTY</th>
<th>UID</th>
<th>STIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>2152</td>
<td>1</td>
<td>2152</td>
<td>con</td>
<td>1000</td>
<td>13:19:07</td>
<td>/usr/bin/bash</td>
</tr>
<tr>
<td>2276</td>
<td>2152</td>
<td>2276</td>
<td>con</td>
<td>1000</td>
<td>14:53:48</td>
<td>/usr/bin/ps</td>
</tr>
</tbody>
</table>

$```

Process group ID

---

**Job Control**

**Introduction**
A *process* is a running program

Some are yours

Most belong to the operating system (or other users)

Use `ps` to get a list

```
$ ps

<table>
<thead>
<tr>
<th>PID</th>
<th>PPID</th>
<th>PGID</th>
<th>TTY</th>
<th>UID</th>
<th>STIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>2152</td>
<td>1</td>
<td>2152</td>
<td>con</td>
<td>1000</td>
<td>13:19:07</td>
<td>/usr/bin/bash</td>
</tr>
<tr>
<td>2276</td>
<td>2152</td>
<td>2276</td>
<td>con</td>
<td>1000</td>
<td>14:53:48</td>
<td>/usr/bin/ps</td>
</tr>
</tbody>
</table>

$ 
```

What terminal (TTY) is it running in?
A *process* is a running program

Some are yours

Most belong to the operating system (or other users)

Use `ps` to get a list

```
$ ps

<table>
<thead>
<tr>
<th>PID</th>
<th>PPID</th>
<th>PGID</th>
<th>TTY</th>
<th>UID</th>
<th>STIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>2152</td>
<td>1</td>
<td>2152</td>
<td>con</td>
<td>1000</td>
<td>13:19:07</td>
<td>/usr/bin/bash</td>
</tr>
<tr>
<td>2276</td>
<td>2152</td>
<td>2276</td>
<td>con</td>
<td>1000</td>
<td>14:53:48</td>
<td>/usr/bin/ps</td>
</tr>
</tbody>
</table>

$ 
```

What terminal (TTY) is it running in? '?' indicates a system service (no TTY)
A process is a running program
Some are yours
Most belong to the operating system (or other users)
Use `ps` to get a list

```
$ ps

<table>
<thead>
<tr>
<th>PID</th>
<th>PPID</th>
<th>PGID</th>
<th>TTY</th>
<th>UID</th>
<th>STIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>2152</td>
<td>1</td>
<td>2152</td>
<td>con</td>
<td>1000</td>
<td>13:19:07</td>
<td>/usr/bin/bash</td>
</tr>
<tr>
<td>2276</td>
<td>2152</td>
<td>2276</td>
<td>con</td>
<td>1000</td>
<td>14:53:48</td>
<td>/usr/bin/ps</td>
</tr>
</tbody>
</table>

$```

The user ID of the process's owner
A *process* is a running program
Some are yours
Most belong to the operating system (or other users)
Use `ps` to get a list

```
$ ps

PID    PPID   PGID   TTY   UID     STIME   COMMAND
2152      1   2152   con  1000  13:19:07   /usr/bin/bash
2276   2152   2276   con  1000  14:53:48   /usr/bin/ps

$ 
```

The user ID of the process's owner
Controls what the process can read, write, execute, ...
A process is a running program
Some are yours
Most belong to the operating system (or other users)
Use `ps` to get a list

```
$ ps
PID   PPID   PGID   TTY   UID   STIME   COMMAND
2152   1     2152   con  1000  13:19:07   /usr/bin/bash
2276  2152   2276   con  1000  14:53:48   /usr/bin/ps
$
```

When the process was started
A *process* is a running program
Some are yours
Most belong to the operating system (or other users)
Use `ps` to get a list

$ ps

<table>
<thead>
<tr>
<th>PID</th>
<th>PPID</th>
<th>PGID</th>
<th>TTY</th>
<th>UID</th>
<th>STIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>2152</td>
<td>1</td>
<td>2152</td>
<td>con</td>
<td>1000</td>
<td>13:19:07</td>
<td>/usr/bin/bash</td>
</tr>
<tr>
<td>2276</td>
<td>2152</td>
<td>2276</td>
<td>con</td>
<td>1000</td>
<td>14:53:48</td>
<td>/usr/bin/ps</td>
</tr>
</tbody>
</table>

$ The program the process is executing
Can stop, pause, and resume running processes
Can stop, pause, and resume running processes

$ ./analyze results*.dat
Can stop, pause, and resume running processes

$ ./analyze results*.dat

...a few minutes pass...
Can stop, pause, and resume running processes

$ ./analyze results*.dat

...a few minutes pass...

^C

$
Can stop, pause, and resume running processes

$ ./analyze results*.dat

...a few minutes pass...

^C

Stop the running program

$
Can stop, pause, and resume running processes

$ ./analyze results*.dat
...a few minutes pass...
^C
$ ./analyze results*.dat &
$
Can stop, pause, and resume running processes

$ ./analyze results*.dat
...a few minutes pass...
^C
$ ./analyze results*.dat &
$
Can stop, pause, and resume running processes

$ ./analyze results*.dat
...a few minutes pass...
^C
$ ./analyze results*.dat &
$ 

Run in the *background*
Shell returns right away instead of waiting for the program to finish
Can stop, pause, and resume running processes

$ ./analyze results*.dat
...a few minutes pass...
^C
$ ./analyze results*.dat &
$ fbcmd events
$
Can run other programs in the *foreground*
while waiting for background process(es) to finish
Can stop, pause, and resume running processes

$ ./analyze results*.dat

...a few minutes pass...

^C

$ ./analyze results*.dat &

$ fbcmd events

$ jobs

[1] ./analyze results01.dat results02.dat results03.dat

$
Can stop, pause, and resume running processes

$ ./analyze results*.dat
... a few minutes pass...
^C
$ ./analyze results*.dat &
$ fbcmd events
$ jobs  Show background processes
[1] ./analyze results01.dat results02.dat results03.dat
$
Can stop, pause, and resume running processes

```
$ ./analyze results*.dat
...a few minutes pass...
^C
$ ./analyze results*.dat &
$ fbcmd events
$ jobs
[1] ./analyze results01.dat results02.dat results03.dat
$ fg
```
Can stop, pause, and resume running processes

```
$ ./analyze results*.dat
...a few minutes pass...
^C
$ ./analyze results*.dat &
$ fbcmd events
$ jobs
[1] ./analyze results01.dat results02.dat results03.dat
$ fg
```

Bring background job to foreground
Can stop, pause, and resume running processes

$ ./analyze results*.dat

...a few minutes pass...

^C

$ ./analyze results*.dat &

$ fbcmd events

$ jobs

[1] ./analyze results01.dat results02.dat results03.dat

$ fg

Bring background job to foreground

Use fg %1, fg %2, etc. if there are several background jobs
Can stop, pause, and resume running processes

$ ./analyze results*.dat
...a few minutes pass...
^C
$ ./analyze results*.dat &
$ fbcmd events
$ jobs
[1] ./analyze results01.dat results02.dat results03.dat
$ fg
...a few minutes pass...

$ And finally it's done
Use ^Z to pause a program that's already running
Use ^Z to pause a program that's already running

fg to resume it in the foreground
Use ^Z to pause a program that's already running

fg to resume it in the foreground

Or bg to resume it as a background job
Use ^Z to pause a program that's already running
fg to resume it in the foreground
Or bg to resume it as a background job

$ ./analyze results01.dat
Use ^Z to pause a program that's already running
fg to resume it in the foreground
Or bg to resume it as a background job

$ ./analyze results01.dat
^Z
[1] Stopped ./analyze results01.dat
$
Use \texttt{\textasciitilde Z} to pause a program that's already running \texttt{fg} to resume it in the foreground
Or \texttt{bg} to resume it as a background job

$ ./analyze results01.dat$

\texttt{\textasciitilde Z}

\texttt{[1] Stopped ./analyze results01.dat}

$ bg %1$

$
Use `^Z` to pause a program that's already running

`fg` to resume it in the foreground

Or `bg` to resume it as a background job

```
$ ./analyze results01.dat
^Z
[1]  Stopped  ./analyze results01.dat
$  bg  %1
$  jobs
[1]  ./analyze results01.dat
$
```
Use ^Z to pause a program that's already running
fg to resume it in the foreground
Or bg to resume it as a background job

$ ./analyze results01.dat
^Z
[1] Stopped ./analyze results01.dat
$ bg %1
$ jobs
[1] ./analyze results01.dat
$ kill %1
$
Job control mattered a lot when users only had one terminal window.
Job control mattered a lot when users only had one terminal window

Less important now: just open another window
Job control mattered a lot when users only had one terminal window
Less important now: just open another window
Still useful when running programs remotely