Learning Objectives: Job Control

Learners will be able to...
- Use the bg, fg, and jobs commands.
- Explain what the following key sequences will do - CTRL-Z, CTRL-C, CTRL-D.
- View a list of jobs and a list of processes and explain the difference between the two.
- Use the wait and disown commands.

Types of processes

Two types of computer processes

- Foreground processes
- Background processes

Foreground processes

A foreground process is different from a background process in 2 ways:

1. Some foreground processes show an interface, through which the user can interact with the program. In Linux the command line interface (CLI) is that user interface.
2. The user must wait for one foreground process to complete before running another one.

Background processes

Unlike with a foreground process, the shell does not have to wait for a background process to end before it can run more processes.

You can enter as many background commands one after another within the limit of the amount of memory available.

Try this out:

```
sleep 30
```

You cannot type another command in until the `sleep 30` command is complete.

Try this variation:

```
sleep 30 &
ps
```

The `&` ampersand specifies that the command should run in the background and you have access to the command prompt right away. The `ps` command will list the running processes and you can see that `sleep` is one of them.

Job versus Process

As we learned in the last assignment in this module, a process is a program that is running on your computer. For example, the command `ls` is a process.

Sometimes you may combine two processes in one command line by piping the output of one to the input of the next. These combined processes are called a job.

Example:

```
grep Anne AnneofGreenGables.txt | sort &
```

▼ **What does this command do?**
This command shows you a sorted list of all the lines that contain the word Anne in the file AnneofGreenGables.txt. The `grep` command returns every occurrence of the word `Anne` and that is piped to the `sort` command which returns a sorted list. The two processes in this list are `grep` and `sort` but they are run as one job. The ampersand at the end causes this job to run in the background and it prints a notification that the job is done after you type retunr/enter.

info

A job is one or more processes that are run as part of a single command.

View the running jobs

Set up a couple of running jobs and pause them so we can view a list of jobs. **CTRL-Z** pauses a job.

1. Paste the following command into the terminal and type `CTRL-Z` while it's running:
   `bash find / -name "*.conf" 2>/dev/null`
   ———-

   ▼

   Why is the `2>/dev/null` at the end of the command?

The `2>/dev/null` the `2` means stderr and the `/dev/null` is the "null device" which means throw it away. Adding this filters out the "Permission Denied" errors.

Try it without the error filter:
`find / -name "*.conf"`

2. Paste the following command into the terminal and type `CTRL-Z` while it's running:

```
ls -R /
```

3. Type `jobs` at the prompt.

   You should see something like the following:

   ```
   codio@oreganonull-diegotrivial:~/workspace$ jobs
   [1]-  Stopped                 ls --color=auto -R /
   [2]+  Stopped                 find / -name "*.conf" 2> /dev/null
   ```

   A listing of paused jobs. There are two, one is the find command and the other is ls

4. Type `ps` at the prompt. The list looks a little different, it shows each currently running process, including the `ps` that is outputting the list of processes.

   You should see something like the following:

   ```
   codio@oreganonull-diegotrivial:~/workspace$ ps
     PID TTY          TIME CMD
     692 pts/2    00:00:00 bash
     775 pts/2    00:00:00 find
     782 pts/2    00:00:00 ls
     789 pts/2    00:00:00 ps
   ```

Job Control commands

Linux provides the following commands to control running jobs:

Command	Description
`bg`	Resume the specified job in the background
`fg`	Resume the specified job in the foreground
`jobs`	List all current jobs
`kill`	Can be used to kill a job or a process
`wait`	Wait until the specified job is complete
`disown`	Removes the specified job from the table of active jobs

The `bg` command

This is useful for activities that might be tying up your interface for too long. You can suspend a process using `Ctrl-Z` and then send it to the background. If you don't specify a job number to the `bg` command, it defaults to working on the most recently stopped process.

```
(sleep 30; printf "I am awake\n")
```

- Type `Ctrl- Z` to suspend execution of the job.
- Type `jobs` to view the status
- Type `bg %1` to resume the job
- Type `jobs` to see that it is running again, in the background

The `fg` command

You can use the `fg` command to move a job to the foreground, it will operate on the most recently backgrounded process if you don't specify a job number.

Start the following job in the background:

```
COUNTER=0; while true; do printf "Hello World %d\n" $COUNTER >>
        temp; sleep 1; let COUNTER++; done &
```

▼ **What does this command do?**

This command first initializes a `COUNTER` to 0, then sets up a **forever** loop `while true`, it prints `Hello World` followed by the value of `COUNTER` in the current iteration, then it `sleeps` for a second (to slow it down) and

increments `COUNTER`. The `done` marks the end of the loop.

- Type `cat temp` at any time to see how many iterations it has gone through.
- Type `fg` at the command line to see that it ties up the interface when it runs in the command line. At any point you can type `CTRL-Z` to stop execution and then `cat temp` to see how many more iterations it went through.
- Type `jobs` to see that job is stopped in the queue

The `kill` command for jobs

In the previous assignment we learned about the `kill` command to kill a process. You can also use the `kill` command to kill a job by using the `%` key before the job number.

Type:

```
kill %1
jobs
```

You can see that you have cleared the job queue with the `kill` command.

Sending a signal to a running job

The `CTRL-Z` command

As we have seen in earlier examples, you can use the `CTRL-Z` command to suspend a foreground process. You need to suspend a foreground job in order to have access to the command line. Once you have access to the terminal you can do other things such as send the job to the background `bg` or `kill` it. The `CTRL-Z` command does not work on background processes.

The `CTRL-C` command

You can use `CTRL-C` to kill a foreground process. To kill a background job, you must first bring it into the foreground `fg` and then type `CTRL-C`. It is simpler to use the `kill` command on the process. As a reminder, you can get a list of the running jobs with the `jobs` command.

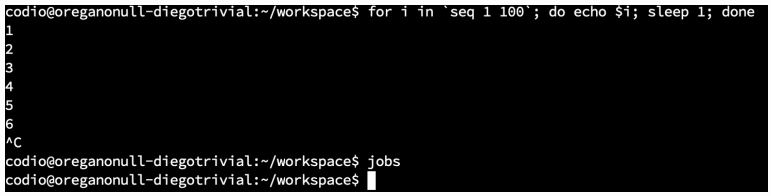
Compare the outcomes of these commands

Run this command:

```
for i in `seq 1 100`; do echo $i; sleep 1; done
```

▼ **What does this command do?**
This sets up a loop that runs from 1 to 100, then it `sleeps` for a second (to slow it down). The `done` marks the end of the loop.

While this is running type `CTRL-C` and then `jobs` to see which jobs are running. You will not see any jobs in the list.

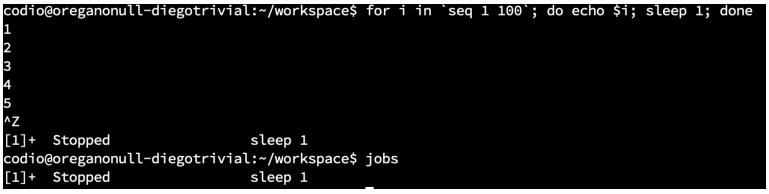


After typing CTRL-C you see that the jobs queue is empty

Run it again:

```
for i in `seq 1 100`; do echo $i; sleep 1; done
```

Now type `CTRL-Z` and then `jobs` to see which jobs are running. With `CTRL-Z` the job is still there but stopped.



After typing CTRL-Z you see that the jobs queue has one stopped job

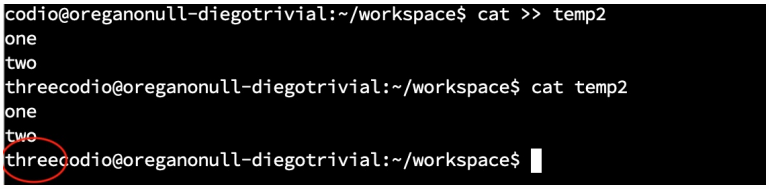
The `CTRL-D` command

The `CTRL-D` command signals the end of input. You can use it to let a command know there will be no more input coming.

Try this:

```
cat >> temp2
```

Enter a few lines and then type `CTRL-D` twice. You have to type it twice if you are not at the end of a line when you type it. If you then type out the contents of the temp2 file `cat temp2` you will see that everything you typed in is there.

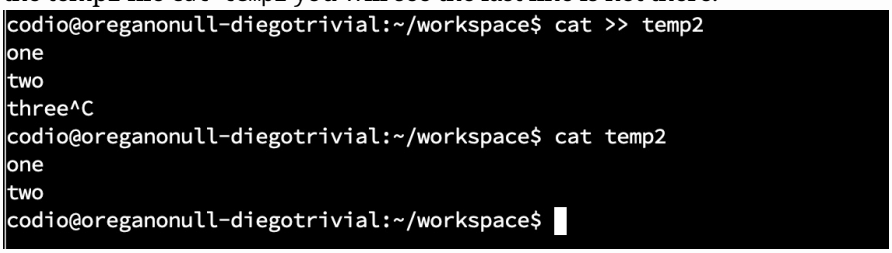


You can see everything you typed in when you end input with CTRL-D

Try this variation:

```
rm temp2
cat >> temp2
```

Before hitting enter on the last line, type `CTRL-C`. Type out the contents of the temp2 file `cat temp2` you will see the last line is not there.



Other Job Control commands

The `wait` command

This command will suspend script execution until all jobs running in the background have terminated, or if a specific job or process number is specified it will wait until that terminates. Returns the exit status of the waited-for command.

You can use the `wait` command to prevent a script from running before a background job finishes executing.

1. Start this process in the background, it creates a file with the numbers 1 to 20.

```
for i in `seq 1 20`; do echo $i >> temp3; sleep 1; done &
```

2. Enter the command below, if you enter it right away, you'll see that the first job hasn't finished and all the numbers aren't there.

```
cat temp3
```

3. Now enter the command below, the file won't list out until the first job is complete. All the numbers will be there.

```
wait %1; cat temp3
```

The `disown` command

The `disown` command removes a job from the table of active jobs. If you don't specify a job number it will disown the most recently launched job.

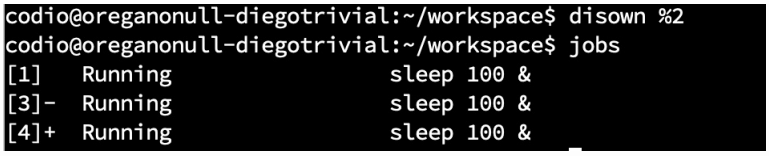
Paste the commands below to set up four jobs running in the background:

```
sleep 100 &
sleep 100 &
sleep 100 &
sleep 100 &
jobs
```

Try this:

```
disown %2
jobs
```

You will see something like below, jobs 1, 3 and 4 remain.



running disown %2 and then jobs show you a list of jobs that contains 1, 3 and 4

▼ **What jobs will still be running if you type in `disown`?**

Typing `disown` without a job number will disown the last job run, what will remain are jobs 1 and 3.

Setting process priority

The `nice` and `renice` commands are used to set and change the priority of a Linux process. The term `nice` refers to the fact that high-priority tasks are less *nice* because they don't share resources as well as low-priority tasks. Priority values range from -20 (highest priority) to 19 (lowest priority) .

Using the `-l` flag with the `ps` command will list the priority of jobs, the priority is in the **NI** column. You can also see the *nice* values in the **NI** column when you use the `top` command.

Try the `ps` command:

```
ps -l
```

Use `nice` to set the priority of a process before it is started

```
nice -n 19 sleep 100 &
```

View the **NI** column:

```
ps -l
```

You can use `renice` to set the priority of an already running process

Learn more about the renice command:

```
man renice
```