Learning Objectives

Learners will be able to...
- Use appropriate system services
- Schedule processes
- Manage processes

Computer Processes

What is a process?

A process is a program that is running on your computer. Linux is a multiprocessing system, each process is a separate task, runs in its own virtual address space and can only interact with other processes through the kernel. This isolation ensures that if a process crashes, it won't bring down other processes.

A process encompasses a program's instructions and data, the program counter, the CPU registers and the process stacks which contain temporary data such as routine parameters, return addresses and saved variables.

We can take a look at the processes running on your Linux virtual machine by typing the command `top` in the terminal window.

Try it out:

```
top
```

The `top` program displays an interactive, live, sorted list of system processes. The default sorting key is **PID** (process identification number), but other keys can be used instead.

In the rightmost column, titled **COMMAND**, you can see the commands that are running.

Now open a new terminal window by clicking the terminal icon in the file tree:

The terminal icon in the file tree is circled. You can also access the terminal from the menu by selecting Tools Terminal

At the prompt in the new terminal window type the following:

```
sleep 30
```

The `sleep 30` command tells the computer to sleep for 30 seconds.

Click back on the first terminal window and you will see that `sleep` has been added to the list of running processes. It should look something like the image below.

You can stop the `top` program by typing the letter `q`

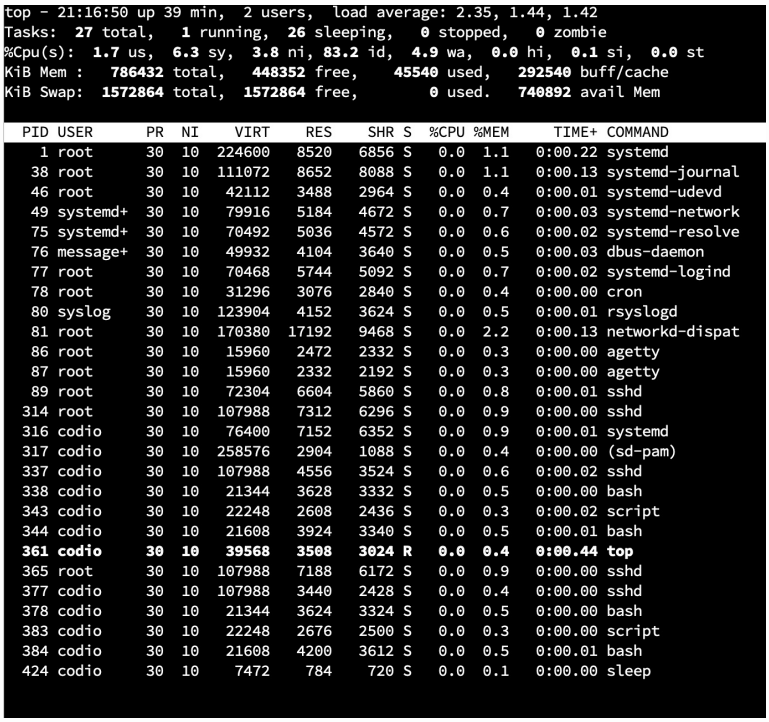

Image of the top process running. You can see top and sleep in the command list.

Information About Running Processes

Interpreting the top command

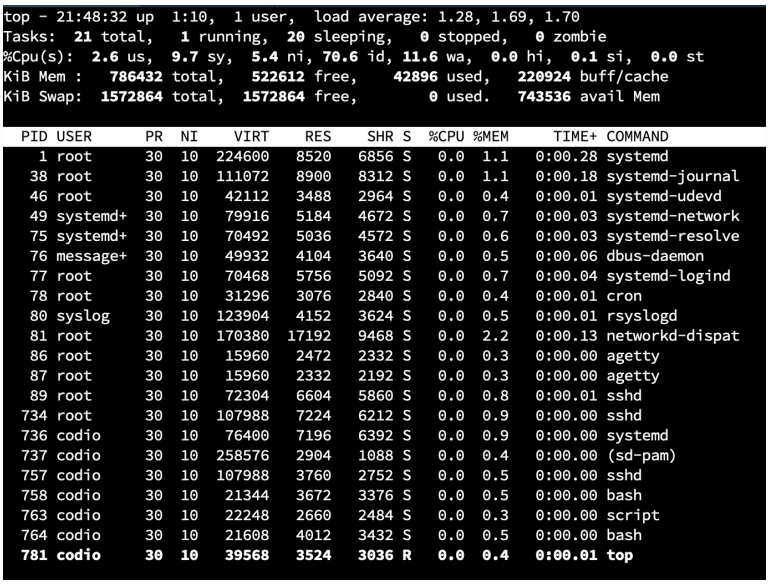

Image of the top process running.

Running the `top` command provides the following information:
- **Tasks**: A list of the number of current processes the computer is running. The `top` program is the one labeled as *running* while other software such as the *bash shell* is paused. *Zombie* programs are the ones that fail to quit or *crash* because of a bug or a system memory overload.

- **PID**: The *process identification number*. *Running* or *Zombie* programs can be *killed* by using this unique id.

- **USER**: The user that started the task. For example, the *root* user tasks, also called *superuser* are running in the background. The *codio* user is running a *bash* shell and the *top* command itself (PID 758 and 781 respectively).

- **%CPU & %MEM**: Display the percentage of the computer's *Central Processing Unit* usage and the *Random Access Memory* or RAM usage. These indicators can help you identify which program is making the computer *laggy* or slow.

- **COMMAND**: The name of the program (software).

1. Run the `top` program again:

```
top
```

2. Press the `shift + f` key to display a list of possible sorting keys. Scroll through the entries to highlight whatever you are interested in sorting by. Type `s` to select it and `ESC` to return to viewing the processes. They will now be sorted by the key you selected.

As was mentioned on the previous page, you can exit the top program by typing the letter `q`. Let's try ending it a different way.

3. Take note of the PID of the `top` program.
4. Open another terminal window and type `kill` followed by the PID of the top program.

info

Other ways to find the PID

If you forgot the PID - either of the following commands will provide it to you:

```
pgrep top
```

```
pidof top
```

The main difference between these two commands is that `pidof` will find processes that are an exact match of the search string and `pgrep` can find processes that contain the string.

5. Click the tab to switch back to the first terminal window and you will see the command prompt, signaling that the top program is no longer running.

Other ways to see running processes

`ps` - Process state

The `ps` command differs from top in that it is not interactive - it prints out the list and exits. You can use `ps` in conjunction with other commands that might do something with the list.

Use the `man` command to view the parameters you can supply to `ps`.

```
man ps
```

`htop` - a snazzier version of top

The `htop` command displays all processes running on the system, along with their command line arguments. You can view the commands in tree format and select multiple processes and act on them at the same time. You don't have to enter the PID to perform tasks related to processes.

Use `q` to quit as the `F10` key maps to something else in Codio.

`lsof` - Lists open files

Try the `lsof` command:

```
lsof
```

|||challenge

How can you make `prideandprejudice.txt` be one of the open files?

▼ Seeing an open file in the `lsof` list

One way to do this is to open a new terminal window and run `more prideandprejudice.txt` then go back to the original terminal window and type `lsof` and see that `more` has that file open.

|||

Process Management

Killing a process

On the previous page we tried out the `kill` command to end the `top` process. There is another version of the kill command where you don't need to supply the PID, just the process name.

Try this out in the terminal window type:

```
sleep 100
```

Open a new terminal window and type:

```
pkill sleep
```

If you then switch back to the first terminal window, you will see **Terminated** under the sleep command.

---

challenge

What do you think would have happened if you had multiple `sleep` processes?

▼ Multiple processes with the same name

All processes with the same name will be killed with the `pkill` process. You can try this out by running sleep in two terminal windows and then in a third terminal window type `pkill sleep`.

Kill signals

All processes should contain code that will handle a kill signal, these are called signal handlers. A typical action of a signal handler might be to delete temporary files or prompt to save changes. The exceptions to this are SIGKILL and SIGSTOP which are immediate and cannot be handled, ignored or blocked.

You can view a list of all the possible signals available with the `kill` command by typing:

```
kill -l
```

As you can see, there are many possible signals. If you do not specify a signal to the `kill` (or `pkill`) command, the default is `SIGTERM` which allows the application to clean up before quitting.

The most commonly used signals are:

Signal Name	Value	Meaning
SIGINT	2	Similar to pressing Ctrl-C, it may be ignored by the process
SIGHUP	1	This is a hang-up signal and is used to report that the user's terminal is disconnected
SIGQUIT	3	Similar to SIGINT but with the ability to produce a core dump
SIGKILL	9	Forces the process to terminate immediately and cannot be ignored
SIGTERM	15	Gives the application time to shutdown gracefully and may be ignored

Viewing a graceful shutdown

We can see that `vim` cleans up temporary files when `kill` is called without a signal number, meaning it uses the default **SIGTERM**.

Open one terminal window and type in:

```
vim mytest
```

Take a look at the file tree on the left, `vim` created a temporary swap file.

In another terminal window type in:

```
pkill vim
```

Look at the file tree now, the temporary file has been deleted.

|||challenge

What do you think will happen if you use SIGKILL on the application `vim`?

Try it out using a similar procedure as above but use this `pkill -9 vim` command instead.

▼ How `vim` handles sudden termination.

The temporary file will not be deleted. This is probably a good thing because if you have made edits and the process gets killed you'll at least have something saved in the swap file.

|||

Process States

In Linux there are five possible states a process may be in:

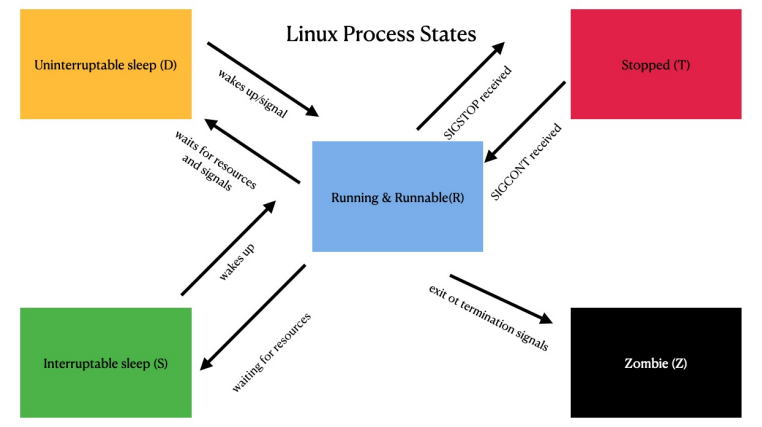

Diagram of the 5 states a process can be in

Processes always start off in the Running or Runnable state. After it starts it might change states to one of the sleeping states if it needs to wait for resources or signals.

Running or Runnable *R*

When the process is running it is using the CPU to execute instructions. It can also be in the runnable state which means it is in the scheduling queue for using the CPU.

Uninterruptible Sleep - *D* / Interruptable Sleep - *S*

A process which needs to wait for resources such as IO or data or network requests will enter sleep mode. This allows other processes to use the CPU while the process waits for the resources. The difference between the two types of sleep states is that the **Interruptable Sleep** state will react to both the availability of the resources it needs or to signals. The **Uninterruptible Sleep** state only reacts to the availability of the resources.

Stopped - *T*

A process will enter the stopped state when it receives either the SIGSTOP or SIGTSTP signals. The SIGSTOP is not ignorable but a process can choose to ignore the SIGTSTP signal. The SIGCONT signal returns the process to a

running state.

Zombie - *Z*

When a process is terminated or has completed it sends a SIGCHLD signal to the parent process and enters a Zombie state. The parent process is responsible for clearing the process from the process table.