Process management in Linux

Linux Process

Management

Daemon process

Orphan process

Zombie process

Parent process

Child process

Table of content

Before starting this lesson, launch and connect remotely to any linux server from your Visual Studio code

Introduction

What is a process?

Introduction

- A **process**, in simple terms, is an **instance of a running program**.
- Whenever you issue a command in **UNIX**, it creates, or starts, a new process.
- When you tried out the **ls** command to list directory contents, **you started a process!**

Process ID

What is a process ID?

Process ID

- The operating system tracks processes through a five digit ID number known as the **PID** or **process ID** .
- Each process in the system has a **unique pid**.
- **PIDs** eventually repeat because all the possible numbers are used up and **the next PID rolls or starts over**.
- At any one time, **no two processes with the same pid exist in the system** because it is the **pid** that UNIX uses to track each process.

Types of processes in Linux

Types of processes

- In Linux processes can be of two types:
  - **Foreground Processes** (require a user to start them or to interact with them) also referred to as **interactive processes**
  - **Background Processes** (run independently of a user. referred to as **non-interactive or automatic processes**
- Programs and commands run as **foreground processes** by default.

1 Starting a process

How do we start a process in Linux?

Starting a process

- By default, every process that you start runs in the foreground.
- It **gets its input from the keyboard** and **sends its output to the screen**.
- And if it need some output from the CLI , it will wait for the user to enter it .
- **Example:** The **ls** command
- If I want to list all the files in my current directory, I can use ls
- It will display the content of the directory on the foreground.

Starting a process

- While a program is running in foreground and taking much time, **we cannot run any other commands** (or start any other process)
- This is because prompt would not be available until the program finishes its processing and comes out.
- But there are some tricks we can use to **stop a process**

Listing running processes

List running processes

- It is easy to see your own processes by running the **ps** command as follows

```
$ps
PID    TTY     TIME      CMD
18358  ttyp3   00:00:00  sh
18361  ttyp3   00:01:31  abiword
18789  ttyp3   00:00:00  ps
```

- **ps** stands for **Process Status**

- One of the most commonly used flags for **ps** is the **-f** (f for full) option
- It provides more information as shown in the following example

```
$ps -f
UID     PID  PPID C STIME    TTY   TIME CMD
amrood  6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood  6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood  3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood  6892 3662 4 10:51:50 pts/6 0:00 ps -f
```

ps commands fields

Here is the description of all the fields displayed by **ps -f** command

Column	Description
UID	User ID that this process belongs to (The person running it)
PID	Process ID
PPID	Parent Process ID (The ID of the process that started it)
C	CPU utilization of Process
STIME	Process start time
TTY	Terminal type associated with the process
TIME	CPU time taken by the process
CMD	Command that started this process

Options of the ps command

There are other options which can be used along with the **ps** command

Option	Description
-a	Shows information about all users
-x	Shows information about processes without terminals
-u	Shows additional information like -f option
-e	Display extended information

The top command

The top command

- The **top** command is a very useful tool for quickly **showing processes sorted by various criteria.**
- It is an **interactive diagnostic tool** that updates frequently and **shows information about physical and virtual memory, CPU usage, load averages, and your busy processes.**
- You can run top command and to see the statistics of CPU utilization by different processes: **$ top**

The top command

- When you run the top command, you can:
  - Press **s** to Enter how long it will refresh then press **Enter** to validate (**3 second is default)**
  - We can put it 1 or 2 seconds
  - Press **I** (**i**) to hide all the **idle processes** and **leave only the ones using the cpu**
  - Press **q** to quit the top command interface

2

Stopping Processes

How to stop a process in Linux?

Stop a process

- **Ending a process can be done in several different ways.**
- Often, from a console-based command, sending a **CTRL + C** keystroke (the default interrupt character) **will exit the command.**
- This **works when process is running in foreground mode.**

Stop a process

- If a process is running in **background mode** then:
- First you would need to get its **Job ID** using **ps** command
- And after that you can use **kill** command to **kill the process** as follows

```
$ps -f
UID      PID  PPID C STIME    TTY   TIME CMD
amrood   6738 3662 0 10:23:03 pts/6 0:00 first_one
amrood   6739 3662 0 10:22:54 pts/6 0:00 second_one
amrood   3662 3657 0 08:10:53 pts/6 0:00 -ksh
amrood   6892 3662 4 10:51:50 pts/6 0:00 ps -f
$kill 6738
Terminated
```

Stop a process

- Here **kill** command would terminate **first_one** process.
- If a process ignores a regular **kill** command, you can use **kill -9** followed by the **process ID** as follows

```
$kill -9 6738
Terminated
```

3

Parent and Child processes

PPID and PID

Parent and Child process

- Each unix process has two ID numbers assigned to it:
  - **Process ID (pid)**
  - **Parent process ID (ppid).**
- **Each user process in the system has a parent process.**
- Most of the commands that you run have the **shell as their parent**.
- Check **ps -f** example where this command listed both **process ID and parent process ID**

4

Zombie and Orphan processes

What is the difference?

Zombie and Orphan Processes

- Normally, when a child process is killed, the parent process is told via a **SIGCHLD signal**.
- Then the parent can do some other task or **restart a new child as needed**.
- However, sometimes **the parent process is killed before its child is killed**.
- In this case, the "parent of all processes," **init process**, becomes the new **PPID (parent process ID).**
- **These processes are called orphan process.**

Zombie and Orphan Processes

- When a process is killed, a **ps** listing may still show the process with a **Z** state.
- This is a **zombie**, or **defunct process**. The process is **dead** and not being used.
- These **processes are different from orphan processes**.
- **They are the processes that has completed execution but still has an entry in the process table.**

5

Daemon processes

What is a daemon in Linux?

Daemon Processes

- Daemons are **system-related background processes** that often run with the permissions of **the root and services requests** from other processes.
- A daemon process has no controlling terminal. It cannot open **/dev/tty**.
- If you do a **ps -ef** and look at the **tty** field, all daemons will have a **?** for the **TTY**

Daemon Processes

- More clearly, **a daemon is just a process that runs in the background**
- Usually waiting for something to happen that it is capable of working with, like **a printer daemon is waiting for print commands**.
- If you have a program which needs to do **long processing** then **its worth to make it a daemon and run it in background.**

6

Job ID vs Process ID

What is the difference?

Job ID vs Process ID

- Background and suspended processes are usually manipulated via a **job number (job ID)**.
- This number is different from the **process ID** and **is used because it is shorter.**
- In addition, **a job can consist of multiple processes running in series or at the same time (in parallel)**
- So **using the job ID is easier than tracking the individual processes.**

Thanks!

Any questions?

You can find us at:

website: http://utrains.org/

Phone: +1 (302) 689 3440

Email: contact@utrains.org

Click on the link below to contact the support team for any issue!

utrains.org/support/

Create a ticket for your problem and we will get back to you soon!