



Process Management



Prepared by:

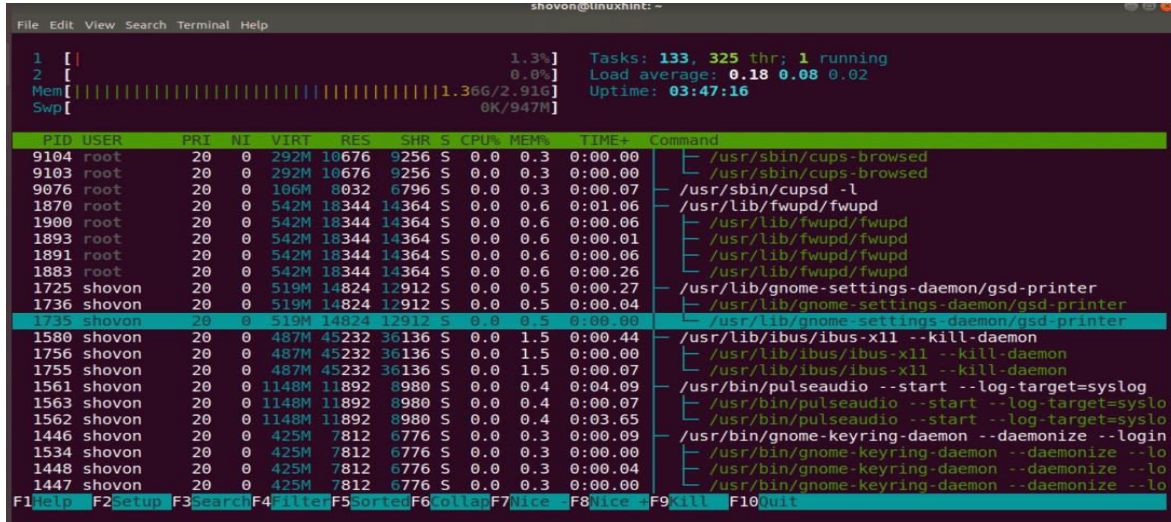
Ali Khoshtinat

Aryan Haddadi

Operating Systems
Spring 1400

View all processes running on Linux live

- In Linux, you can see the system status and information of all processes using **htop** command.



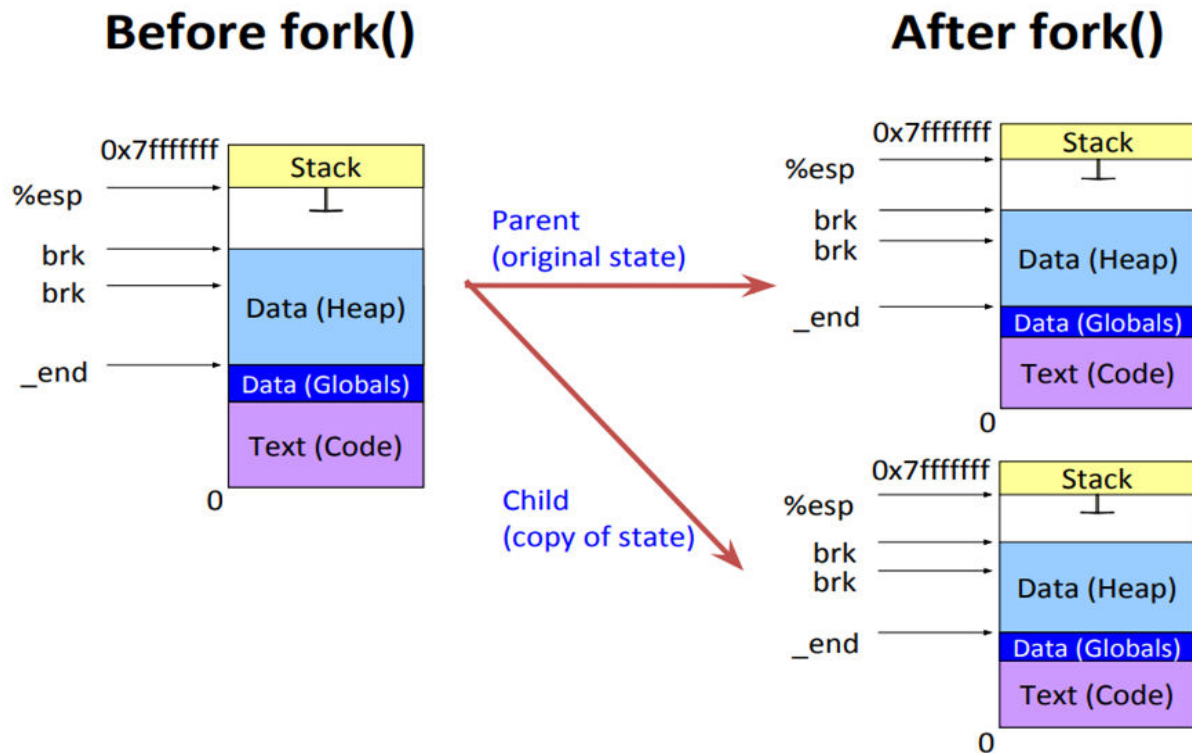
The screenshot shows the htop interface in a terminal window. At the top, it displays system statistics: 1.3% CPU usage, 0.0% memory usage, 1.36G/2.91G memory, and 0K/947M swap. It also shows 133 tasks, 325 threads, and 1 running process. The load average is 0.18, 0.08, 0.02, and the uptime is 03:47:16.

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
9104	root	20	0	292M	10676	9256	S	0.0	0.3	0:00.00	/usr/sbin/cups-browsed
9103	root	20	0	292M	10676	9256	S	0.0	0.3	0:00.00	/usr/sbin/cups-browsed
9076	root	20	0	106M	8032	6796	S	0.0	0.3	0:00.07	/usr/sbin/cupsd -l
1870	root	20	0	542M	18344	14364	S	0.0	0.6	0:01.06	/usr/lib/fwupd/fwupd
1900	root	20	0	542M	18344	14364	S	0.0	0.6	0:00.06	/usr/lib/fwupd/fwupd
1893	root	20	0	542M	18344	14364	S	0.0	0.6	0:00.01	/usr/lib/fwupd/fwupd
1891	root	20	0	542M	18344	14364	S	0.0	0.6	0:00.06	/usr/lib/fwupd/fwupd
1883	root	20	0	542M	18344	14364	S	0.0	0.6	0:00.26	/usr/lib/fwupd/fwupd
1725	shovon	20	0	519M	14824	12912	S	0.0	0.5	0:00.27	/usr/lib/gnome-settings-daemon/gsd-printer
1736	shovon	20	0	519M	14824	12912	S	0.0	0.5	0:00.04	/usr/lib/gnome-settings-daemon/gsd-printer
1735	shovon	20	0	519M	14824	12912	S	0.0	0.5	0:00.00	/usr/lib/gnome-settings-daemon/gsd-printer
1580	shovon	20	0	487M	45232	36136	S	0.0	1.5	0:00.44	/usr/lib/ibus/ibus-x11 --kill-daemon
1756	shovon	20	0	487M	45232	36136	S	0.0	1.5	0:00.00	/usr/lib/ibus/ibus-x11 --kill-daemon
1755	shovon	20	0	487M	45232	36136	S	0.0	1.5	0:00.07	/usr/lib/ibus/ibus-x11 --kill-daemon
1561	shovon	20	0	1148M	11892	8980	S	0.0	0.4	0:04.09	/usr/bin/pulseaudio --start --log-target=syslog
1563	shovon	20	0	1148M	11892	8980	S	0.0	0.4	0:00.07	/usr/bin/pulseaudio --start --log-target=syslo
1562	shovon	20	0	1148M	11892	8980	S	0.0	0.4	0:03.65	/usr/bin/pulseaudio --start --log-target=syslo
1446	shovon	20	0	425M	7812	6776	S	0.0	0.3	0:00.09	/usr/bin/gnome-keyring-daemon --daemonize --login
1534	shovon	20	0	425M	7812	6776	S	0.0	0.3	0:00.00	/usr/bin/gnome-keyring-daemon --daemonize --lo
1448	shovon	20	0	425M	7812	6776	S	0.0	0.3	0:00.04	/usr/bin/gnome-keyring-daemon --daemonize --lo
1447	shovon	20	0	425M	7812	6776	S	0.0	0.3	0:00.00	/usr/bin/gnome-keyring-daemon --daemonize --lo

At the bottom, there are function key shortcuts: F1 help, F2 Setup, F3 Search, F4 Filter, F5 Sorted, F6 Collap, F7 Nice, F8 Nice+, F9 Kill, F10 Quit.

- Using this command, you can find every thing about each process and do everything possible with these processes.
- On Ubuntu, you can easily install it using apt command.

Fork



Fork

- **fork()** creates a new process by duplicating the calling process. The new process is referred to as the *child* process. The calling process is referred to as the *parent* process.
- The child process and the parent process run in separate memory spaces. At the time of **fork()** both memory spaces have the same content. Memory writes, file mappings (**mmap()**), and unmappings(**munmap()**) performed by one of the processes do not affect the other.
- The child has its own unique process ID, and this PID does not match the ID of any existing process group (**setpgid()**) or session.

Fork

```
fork (); // Line 1
fork (); // Line 2
fork (); // Line 3
```

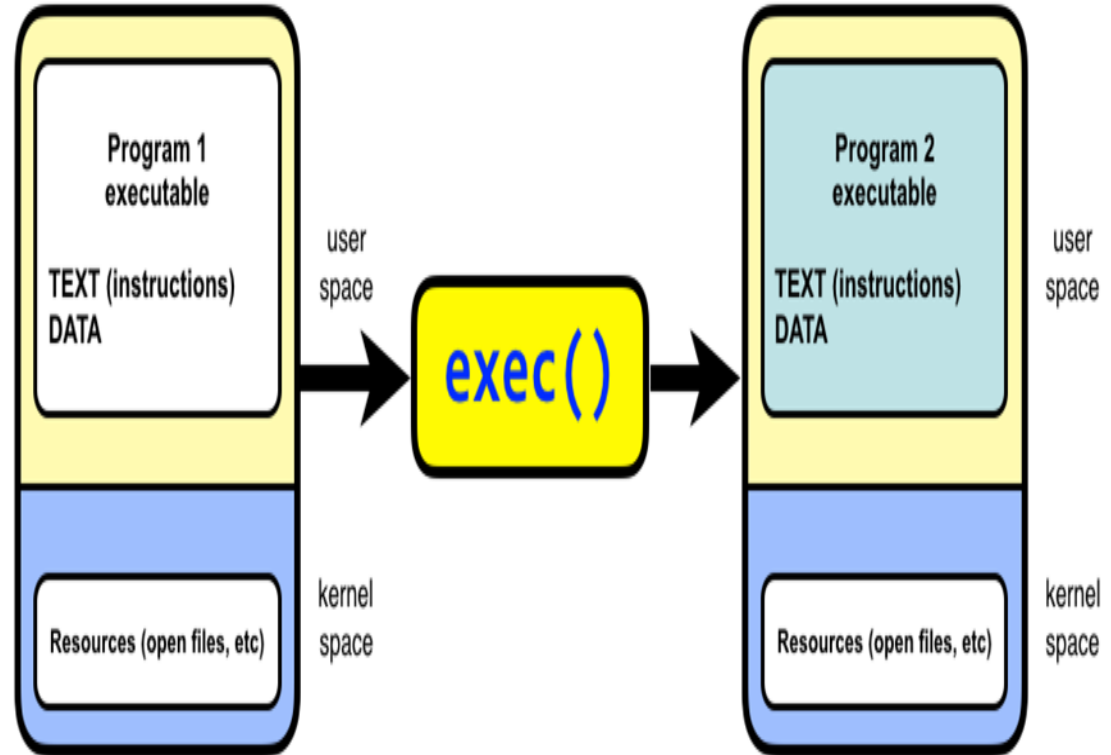
```
      L1      // There will be 1 child process
    /      \   // created by line 1.
  L2      L2   // There will be 2 child processes
 /  \   /  \   // created by line 2
L3  L3  L3  L3 // There will be 4 child processes
```

exec

exec() family of functions or sys calls replaces current process image with new process image.

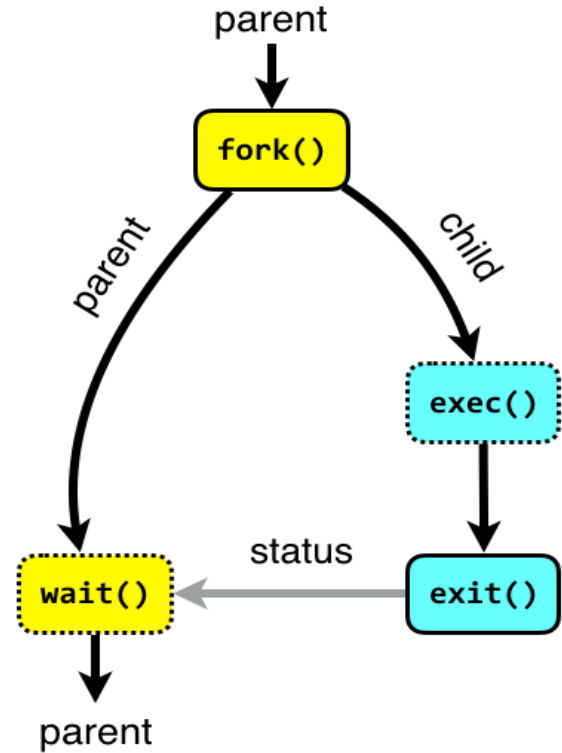
The parent program will be finished after calling exec() even if it still contains other lines of code after the function call.

What to do if we want both?!!



exec + fork

Since the **exec** family of functions **replaces** the current process with a new process image, you need to fork before calling exec, so that the newly forked copy of your process gets replaced (instead of the original getting replaced).



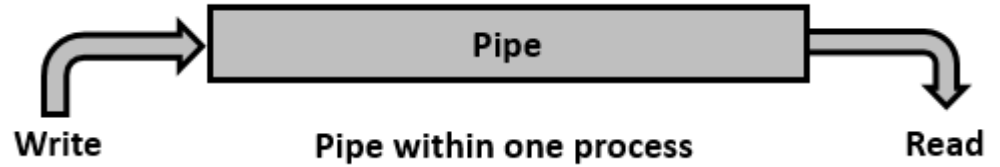
wait

This system call is used to wait for state changes in a child of the calling process, and obtain information about the child whose state has changed. A state change is considered to be: the child terminated; the child was stopped by a signal; or the child was resumed by a signal. In the case of a terminated child, performing a wait allows the system to release the resources associated with the child; if a wait is not performed, then the terminated child remains in a "zombie" state.

As long as a zombie is not removed from the system via a wait, it will consume a slot in the kernel process table, and if this table fills, it will not be possible to create further processes. If a parent process terminates, then its "zombie" children (if any) are adopted by `init()`, which automatically performs a wait to remove the zombies.



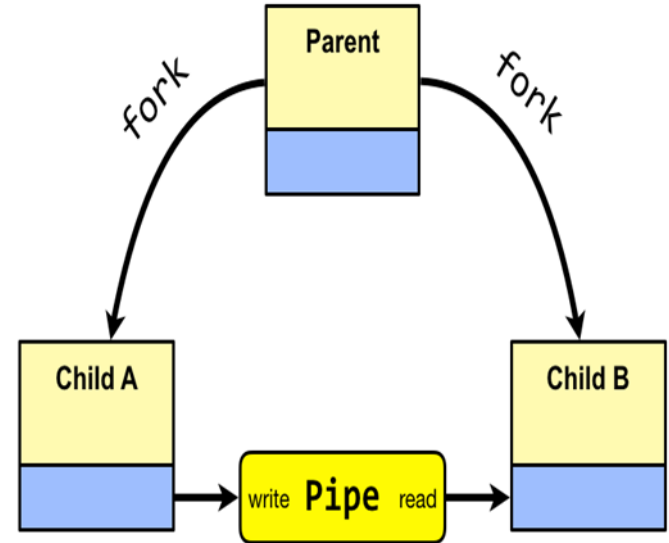
pipe



Pipe is a communication medium between two or more related or interrelated processes. It can be either within one process or a communication between the child and the parent processes. Communication can also be multi-level such as communication between the parent, the child and the grand-child, etc. Communication is achieved by one process writing into the pipe and other reading from the pipe. To achieve the pipe system call, create two files, one to write into the file and another to read from the file.

pipe

- Pipe is one-way communication only i.e we can use a pipe such that One process write to the pipe, and the other process reads from the pipe. It opens a pipe, which is an area of main memory that is treated as a “**virtual file**”.
- The pipe can be used by the creating process, as well as all its child processes, for reading and writing. One process can write to this “virtual file” or pipe and another related process can read from it.
- If a process tries to read before something is written to the pipe, the process is suspended until something is written.



Big Five personality traits

- Openness to Experience
- Conscientiousness
- Extraversion
- Agreeableness
- Neuroticism

Find Personality Difference

- Each user has 5 numbers indicating his/her personality traits.
- You have to find the user with minimum Euclidean distance between his/her personality traits values and each line of traits.csv file.

Euclidean distance

- Euclidean Distance Between 2 Vectors:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

q_i : i -th element of vector q

p_i : i -th element of vector p

n is 5 in this project.