Processes in Linux

Nikita Neveditsin, SMU, 2019

nikita.neveditsin@smu.ca

What is a process?

Process in Linux is just a running program.

ulimit gives information about user limits. In the example max number user processes is set to 15732

cssmuadm@lnx: ~

```
cssmuadm@lnx:~$ cat /proc/sys/kernel/pid_max
32768
cssmuadm@lnx:~$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 15732
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 15732
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
cssmuadm@lnx:~$
```

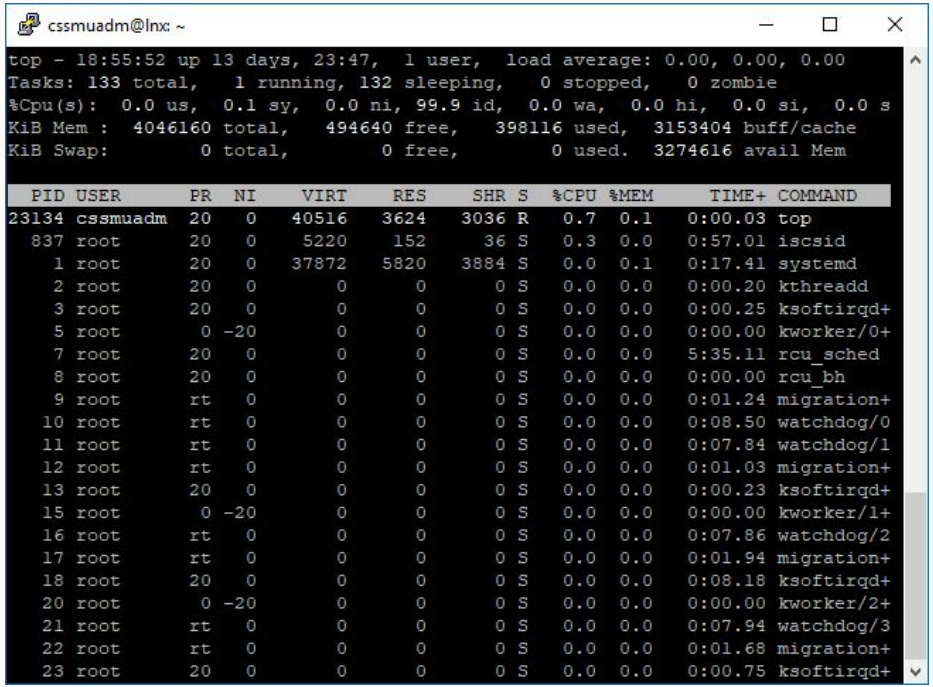
top

top command is used to display Linux processes

Each process has a unique identifier (PID)

There are 4 possible STATES of processes:

- *Running*
- *Sleeping* (waiting for an event/resource)
- *Stopped*
- *Zombie* (dead process, usually indicates resource leak/error if stays for a long time)



```
cssmuadm@lnx: ~
top - 18:55:52 up 13 days, 23:47,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 133 total,   1 running, 132 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.1 sy,  0.0 ni, 99.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 s
KiB Mem :  4046160 total,   494640 free,   398116 used,  3153404 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  3274616 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
23134 cssmuadm  20   0   40516   3624   3036 R   0.7  0.1   0:00.03 top
  837 root      20   0    5220    152     36 S   0.3  0.0   0:57.01 iscsid
    1 root      20   0   37872   5820   3884 S   0.0  0.1   0:17.41 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.20 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   0:00.25 ksoftirqd+
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0+
    7 root      20   0       0      0      0 S   0.0  0.0   5:35.11 rcu_sched
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      rt   0       0      0      0 S   0.0  0.0   0:01.24 migration+
   10 root      rt   0       0      0      0 S   0.0  0.0   0:08.50 watchdog/0
   11 root      rt   0       0      0      0 S   0.0  0.0   0:07.84 watchdog/1
   12 root      rt   0       0      0      0 S   0.0  0.0   0:01.03 migration+
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.23 ksoftirqd+
   15 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/1+
   16 root      rt   0       0      0      0 S   0.0  0.0   0:07.86 watchdog/2
   17 root      rt   0       0      0      0 S   0.0  0.0   0:01.94 migration+
   18 root      20   0       0      0      0 S   0.0  0.0   0:08.18 ksoftirqd+
   20 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/2+
   21 root      rt   0       0      0      0 S   0.0  0.0   0:07.94 watchdog/3
   22 root      rt   0       0      0      0 S   0.0  0.0   0:01.68 migration+
   23 root      20   0       0      0      0 S   0.0  0.0   0:00.75 ksoftirqd+
```

Exercise

Use SHIFT+F for field management in top. Sort processes by used memory

pstree

- Processes in Linux may be represented as a tree with root process init (systemd in modern versions) with *PID = 1*

```
cssmuadm@lnx: ~
cssmuadm@lnx:~$ ls -lh /sbin/init
lrwxrwxrwx 1 root root 20 Mar  8 17:51 /sbin/init -> /lib/systemd/systemd
cssmuadm@lnx:~$
```

- Child processes are “spawned” from parent processes using *fork()* and *exec()* system calls

```
cssmuadm@lnx: ~
cssmuadm@lnx:~$ pstree
systemd─┬─accounts-daemon─┬─{gdbus}
        │                └─{gmain}
        ├─acpid
        ├─agetty
        ├─apache2───6*[apache2]
        ├─atd
        ├─cron
        ├─dbus-daemon
        ├─irqbalance
        ├─2*[iscsid]
        ├─java───32*[{java}]
        ├─lvmetad
        ├─lxcfs───5*[{lxcfs}]
        ├─mdadm
        ├─mysqld───28*[{mysqld}]
        ├─polkitd─┬─{gdbus}
        │         └─{gmain}
        ├─rsyslogd─┬─{in:imklog}
        │          ├─{in:imuxsock}
        │          └─{rs:main Q:Reg}
        ├─snapd───9*[{snapd}]
        ├─sshd─┬─sshd───bash───su───bash
        │      └─sshd───bash───su───bash───pstree
        ├─systemd───(sd-pam)
        ├─systemd-journal
        ├─systemd-logind
        ├─systemd-timesyn───{sd-resolve}
        └─systemd-udevd
cssmuadm@lnx:~$
```

ps

ps command is also used to display Linux processes (a snapshot of the current processes) - can be used in scripts

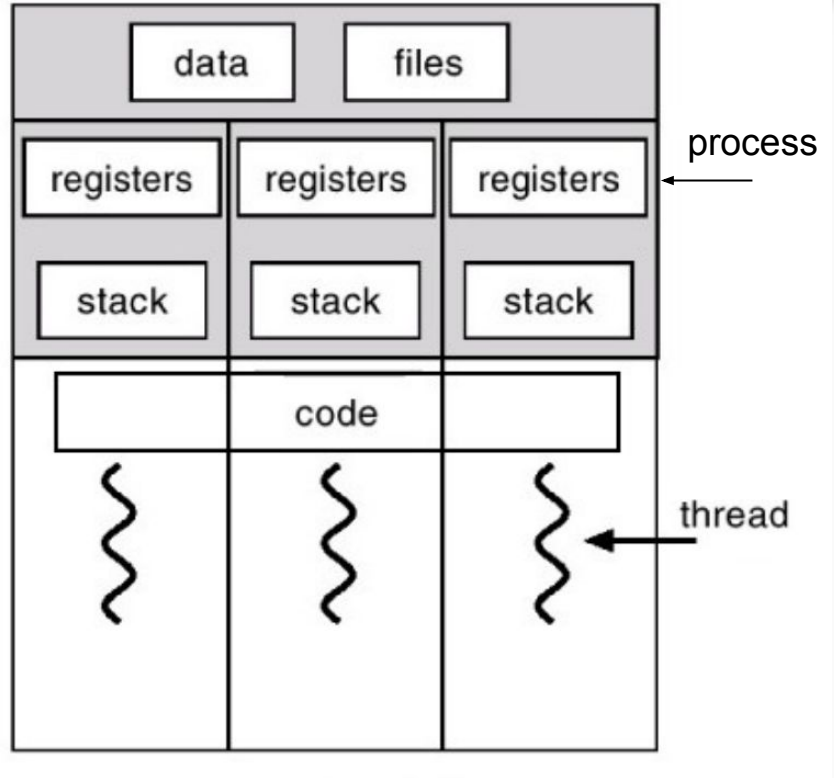
- With no flags - just processes attached to the current terminal
- -ef (UNIX style) or aux (BSD style) - display processes of all users

```
cssmuadm@lnx:~$ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  37872  5820 ?        Ss   May01   0:18 /sbin/init
root         2  0.0  0.0      0     0 ?        S    May01   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    May01   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   May01   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    May01   6:19 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S    May01   0:00 [rcu_bh]
root         9  0.0  0.0      0     0 ?        S    May01   0:01 [migration/0]
root        10  0.0  0.0      0     0 ?        S    May01   0:09 [watchdog/0]
root        11  0.0  0.0      0     0 ?        S    May01   0:08 [watchdog/1]
root        12  0.0  0.0      0     0 ?        S    May01   0:01 [migration/1]
root        13  0.0  0.0      0     0 ?        S    May01   0:00 [ksoftirqd/1]
root        15  0.0  0.0      0     0 ?        S<   May01   0:00 [kworker/1:0H]
root        16  0.0  0.0      0     0 ?        S    May01   0:08 [watchdog/2]
root        17  0.0  0.0      0     0 ?        S    May01   0:02 [migration/2]
root        18  0.0  0.0      0     0 ?        S    May01   0:09 [ksoftirqd/2]
root        20  0.0  0.0      0     0 ?        S<   May01   0:00 [kworker/2:0H]
root        21  0.0  0.0      0     0 ?        S    May01   0:09 [watchdog/3]
root        22  0.0  0.0      0     0 ?        S    May01   0:01 [migration/3]
root        23  0.0  0.0      0     0 ?        S    May01   0:00 [ksoftirqd/3]
root        25  0.0  0.0      0     0 ?        S<   May01   0:00 [kworker/3:0H]
root        26  0.0  0.0      0     0 ?        S    May01   0:00 [kdevtmpfs]
root        27  0.0  0.0      0     0 ?        S<   May01   0:00 [netns]
root        28  0.0  0.0      0     0 ?        S<   May01   0:00 [perf]
root        29  0.0  0.0      0     0 ?        S    May01   0:01 [khungtaskd]
root        30  0.0  0.0      0     0 ?        S<   May01   0:00 [writeback]
root        31  0.0  0.0      0     0 ?        SN   May01   0:00 [ksmd]
```

```
cssmuadm@lnx:~$ ps
  PID TTY          TIME CMD
26086 pts/1    00:00:00 bash
29736 pts/1    00:00:00 ps
cssmuadm@lnx:~$
```

Processes vs threads

- Each process has at least one thread
- Threads in one process share address space (but they have separate stacks and registers), code, and OS resources like open files
- Threads sometimes referred to as “lightweight processes”
- Threads in Linux are implemented in the same way as processes (they both called “tasks” in kernel)



1

Processes vs threads

top -H or ps -T: view threads

cssmuadm@lnx: ~

```
top - 16:47:18 up 15 days, 21:38,  1 user,  load average: 0.00, 0.00, 0.00
Tasks: 134 total,   1 running, 133 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.1 us,  0.1 sy,  0.0 ni, 99.8 id,  0.0 wa,  0.0 hi,  0.0 si,  0.1 s
KiB Mem :  4046160 total,   798684 free,   399752 used,  2847724 buff/cache
KiB Swap:        0 total,        0 free,        0 used.  3272472 avail Mem

  PID USER      %CPU %MEM COMMAND                                  PGRP nTH  TGID
 1069 tomcat8    0.3  4.9 java                                     1045  33  1069
  881 mysql      0.0  3.9 mysqld                                    881  29   881
  725 root       0.0  0.6 snapd                                     725  10   725
  727 root       0.0  0.1 lxcfs                                     727   6   727
  737 syslog     0.0  0.1 rsyslogd                                  737   4   737
  712 root       0.0  0.1 accounts-daemon                           712   3   712
  796 root       0.0  0.1 polkitd                                   796   3   796
  551 systemd+   0.0  0.1 systemd-timesyn                           551   2   551
    1 root       0.0  0.1 systemd                                     1   1     1
    2 root       0.0  0.0 kthreadd                                    0   1     2
    3 root       0.0  0.0 ksoftirqd/0                                0   1     3
    5 root       0.0  0.0 kworker/0:0H                               0   1     5
    7 root       0.0  0.0 rcu_sched                                  0   1     7
    8 root       0.0  0.0 rcu_bh                                     0   1     8
    9 root       0.0  0.0 migration/0                                0   1     9
   10 root       0.0  0.0 watchdog/0                                 0   1    10
   11 root       0.0  0.0 watchdog/1                                 0   1    11
   12 root       0.0  0.0 migration/1                                0   1    12
   13 root       0.0  0.0 ksoftirqd/1                                0   1    13
   15 root       0.0  0.0 kworker/1:0H                               0   1    15
   16 root       0.0  0.0 watchdog/2                                 0   1    16
   17 root       0.0  0.0 migration/2                                0   1    17
   18 root       0.0  0.0 ksoftirqd/2                                0   1    18
   20 root       0.0  0.0 kworker/2:0H                               0   1    20
   21 root       0.0  0.0 watchdog/3                                 0   1    21
   22 root       0.0  0.0 migration/3                                0   1    22
```

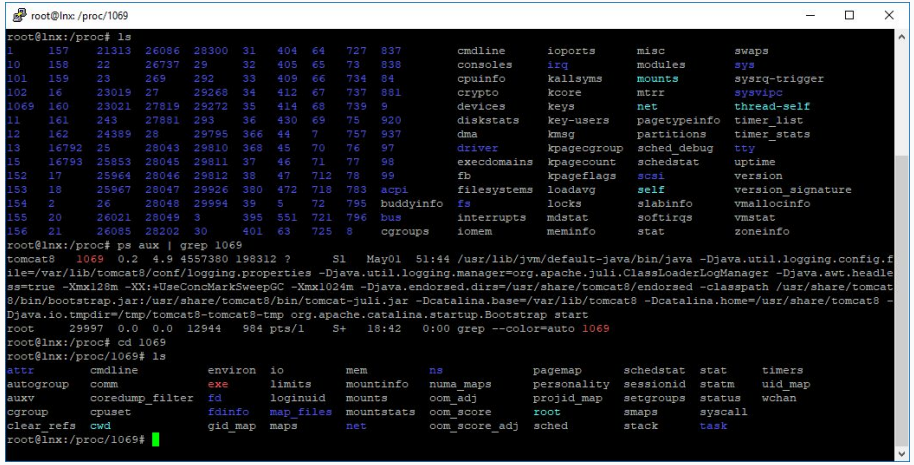
cssmuadm@lnx: ~

```
cssmuadm@lnx:~$ ps -T 1069
  PID  SPID TTY      STAT   TIME COMMAND
 1069  1069 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1177 ?        Sl     0:01 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1191 ?        Sl     0:08 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1192 ?        Sl     0:08 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1193 ?        Sl     0:08 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1194 ?        Sl     0:08 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1195 ?        Sl     4:51 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1218 ?        Sl     1:42 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1219 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1220 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1243 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1244 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1245 ?        Sl     0:18 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1246 ?        Sl     0:20 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1247 ?        Sl     0:14 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1248 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1249 ?        Sl    36:18 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1264 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1265 ?        Sl     1:18 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1351 ?        Sl     2:38 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1352 ?        Sl     1:26 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1353 ?        Sl     1:31 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  1354 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  2855 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  2856 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  7258 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  7259 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  7260 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  7261 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  7262 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
 1069  7263 ?        Sl     0:00 /usr/lib/jvm/default-java/bin/java -Djava.util.lo
```

Processes: /proc directory

/proc is a “pseudo” file system. It contains runtime system information

- The numbered directories contain information about processes where numbers = PIDs



/proc

- cmdline - command line arguments;
- cwd - link to the current working directory;
- environ - list of environment variables;
- exe - link to the executable of this process;
- fd - directory, which contains all file descriptors;
- status - process status in human readable form;

root@lnx: /proc/1069

```
root@lnx:/proc/1069# ls
attr        comm            fd        map_files   net            pagemap      sessionid  status
autogroup   coredump_filter fdinfo    maps        ns             personality  setgroups  syscall
auxv        cpuset          gid_map   mem         numa_maps      projid_map   smaps      task
cgroup      cwd             io        mountinfo   oom_adj        root         stack      timers
clear_refs  environ         limits    mounts      oom_score      sched        stat       uid_map
cmdline     exe             loginuid  mountstats  oom_score_adj  schedstat    statm      wchan
root@lnx:/proc/1069# ls -l cwd
lrwxrwxrwx 1 tomcat8 tomcat8 0 May  8 22:28 cwd -> /var/lib/tomcat8
root@lnx:/proc/1069# cat environ
SHLVL=1OLDPWD=/tmp/tomcat8-tomcat8-tmpHOME=/usr/share/tomcat8TOMCAT8_GROUP=tomcat8TOMCAT8_USER=tom
cat8CATALINA_HOME=/usr/share/tomcat8CATALINA_PID=/var/run/tomcat8.pidJSSE_HOME=/usr/lib/jvm/defaul
t-java/jre/_=/usr/share/tomcat8/bin/catalina.shCATALINA_TMPDIR=/tmp/tomcat8-tomcat8-tmpPATH=/bin:/
usr/bin:/sbin:/usr/sbinJAVA_OPTS=-Djava.awt.headless=true -Xmx128m -XX:+UseConcMarkSweepGCLANG=_SY
STEMCTL_SKIP_REDIRECT=truePWD=/var/lib/tomcat8JAVA_HOME=/usr/lib/jvm/default-javaCATALINA_BASE=/va
r/lib/tomcat8CATALINA_OPTS= -Xmx1024mroot@lnx:/proc/1069#
root@lnx:/proc/1069# cat sta
stack   stat    statm   status
root@lnx:/proc/1069# cat status
Name:   java
State:  S (sleeping)
Tgid:   1069
Ngid:   0
Pid:    1069
PPid:   1
TracerPid:      0
Uid:    113     113     113     113
Gid:    117     117     117     117
FDSize: 128
Groups: 117
NStgid: 1069
```

Exercise

1) Find process id of your **bash** process
2) Using /proc pseudo file system find where link to the executable of this process points to?

Processes: interprocess communication

- Signals
- Pipes
- Named pipes
- Sockets
- Files
- *Shared memory*
- *Message queues*

Processes: IPC: signals

- Signals are typically used in UNIX-like systems (since 1970s) as a notification sent to a process or thread about some event
- Operating system interrupts the process when a signal is sent to it to deliver the signal
- If there is a registered signal handler in the process, it handles the signal. Otherwise, the default signal handler is executed

Processes: IPC: signals

Signal	Value	Action	Comment
SIGHUP	1	Term	Hangup detected on controlling terminal or death of controlling process
SIGINT	2	Term	Interrupt from keyboard
SIGQUIT	3	Core	Quit from keyboard
SIGILL	4	Core	Illegal Instruction
SIGABRT	6	Core	Abort signal from abort(3)
SIGFPE	8	Core	Floating point exception
SIGKILL	9	Term	Kill signal
SIGSEGV	11	Core	Invalid memory reference
SIGPIPE	13	Term	Broken pipe: write to pipe with no readers
SIGALRM	14	Term	Timer signal from alarm(2)
SIGTERM	15	Term	Termination signal
SIGUSR1	30,10,16	Term	User-defined signal 1
SIGUSR2	31,12,17	Term	User-defined signal 2
SIGCHLD	20,17,18	Ign	Child stopped or terminated
SIGCONT	19,18,25	Cont	Continue if stopped
SIGSTOP	17,19,23	Stop	Stop process
SIGTSTP	18,20,24	Stop	Stop typed at terminal
SIGTTIN	21,21,26	Stop	Terminal input for background process
SIGTTOU	22,22,27	Stop	Terminal output for background process

The signals **SIGKILL** and **SIGSTOP** cannot be caught, blocked, or ignored.

2

Processes: IPC: signals

sig is a demonstration program that tries to handle signals

root@lnx: ~

```
  GNU nano 2.5.3            File: sig.c

#include<stdio.h>
#include<signal.h>
#include<unistd.h>

void handler(int sig_no);
int main(void)
{
  //register handlers
  signal(SIGINT, handler);
  signal(SIGHUP, handler);
  signal(SIGTERM, handler);

  /*to demonstrate that
    we can't handle them */
  signal(SIGSTOP, handler);
  signal(SIGKILL, handler);

  while(1)
   sleep(1);
  return 0;
}

void handler(int sig_no)
{
  printf("Signal caught: %d\n", sig_no);
}

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text    ^J Justify
^X Exit        ^R Read File   ^\ Replace     ^U Uncut Text  ^T To Spell
```

kill

kill command sends signal to process (PID)

You can specify number of signal after dash.

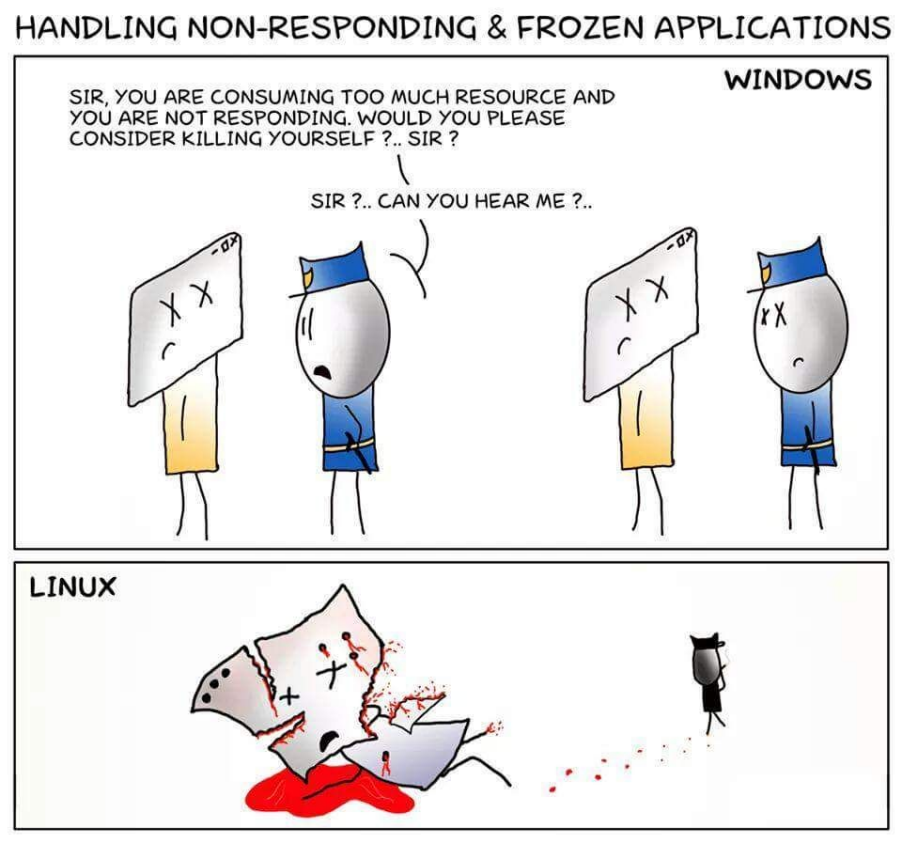
If you do not specify signal the default signal is SIGTERM (15) which CAN BE HANDLED BY PROGRAM (and ignored)

```
cssmuadm@lnx: ~
cssmuadm@lnx:~$ ./sig
Signal caught: 1
^CSignal caught: 2
Signal caught: 2
Signal caught: 15
Killed
cssmuadm@lnx:~$
```

```
cssmuadm@lnx: ~
cssmuadm@lnx:~$ ps aux | grep sig
cssmuadm 18647  0.0  0.0   4220   628 pts/0    S+   14:02   0:00
./sig
cssmuadm 18649  0.0  0.0  11284   936 pts/2    S+   14:02   0:00
grep --color=auto sig
cssmuadm@lnx:~$ kill -1 18647
cssmuadm@lnx:~$ kill -2 18647
cssmuadm@lnx:~$ kill -15 18647
cssmuadm@lnx:~$ kill -9 18647
cssmuadm@lnx:~$
```

SIGKILL

SIGKILL (9) and SIGSTOP (17/9/23) cannot be handled. So, to “kill” a process just send kill -9 to it.

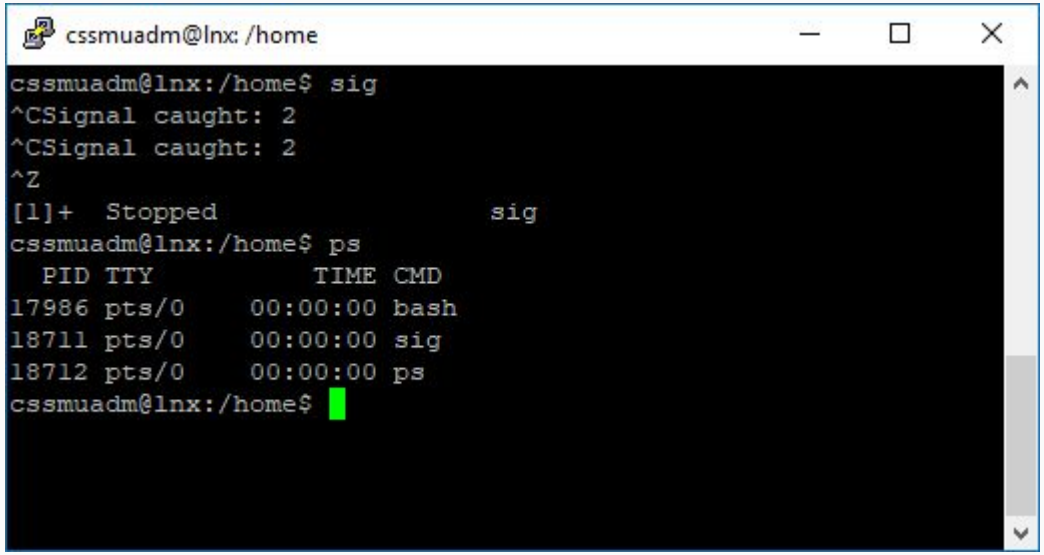


[1]

Processes: IPC: signals

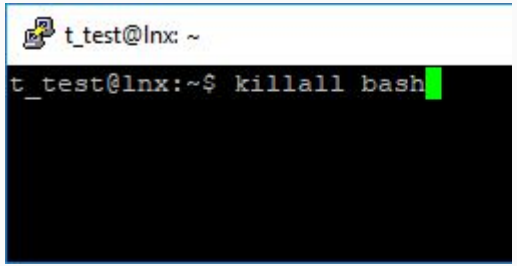
There are 2 important keyboard shortcuts:

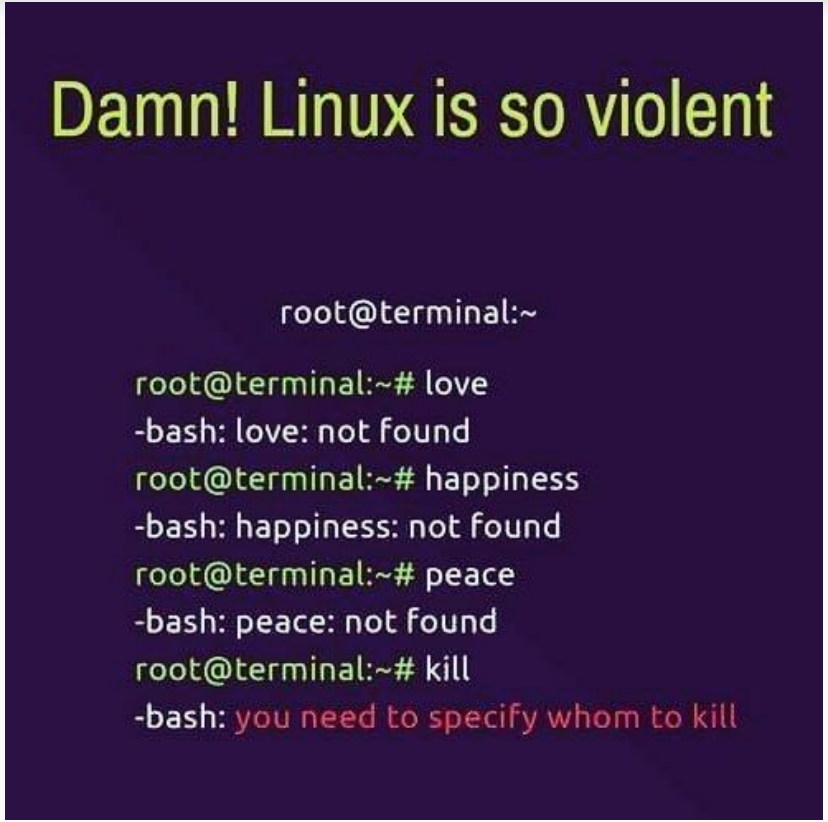
- Ctrl+C sends SIGINT to the current process (if it's not handled by the program then the program terminates)
- Ctrl+Z sends SIGSTOP to the current process. The signal cannot be handled and the process always STOPS (but not terminates)



killall

killall - kill processes by name rather than by process id (PID)





[1]

Exercise

Run **sig** program and terminate it in the current terminal window

Processes: IPC: background processes

There are 2 ways to send a process in background:

- Stop it with Ctrl+Z and then type bg
- Add an ampersand sign (&) at the end of command

NOTE: jobs command shows background processes attached to the current terminal

cssmuadm@lnx: ~

```
cssmuadm@lnx:~$ sig
^Z
[1]+  Stopped                 sig
cssmuadm@lnx:~$ bg
[1]+ sig &
cssmuadm@lnx:~$ jobs
[1]+  Running                 sig &
cssmuadm@lnx:~$ jobs
[1]+  Running                 sig &
cssmuadm@lnx:~$ ps
  PID TTY          TIME CMD
18592 pts/2    00:00:00 bash
18723 pts/2    00:00:00 sig
18724 pts/2    00:00:00 ps
cssmuadm@lnx:~$ kill -9 18723
cssmuadm@lnx:~$ sig &
[2] 18725
[1]   Killed                  sig
cssmuadm@lnx:~$ jobs
[2]+  Running                 sig &
cssmuadm@lnx:~$
```

nohup

If you CLOSE a terminal window, a kernel sends SIGHUP to all processes running in the terminal window (including background processes). Usually programs do not handle this signal and the default action for the signal is process termination.

To avoid process termination use **nohup** command (and send it to background with &)

Processes: IPC: background processes

cssmuadm@lnx: ~

```
cssmuadm@lnx:~$ nohup ./program.sh &
[1] 18959
cssmuadm@lnx:~$ nohup: ignoring input and appending output to 'nohup.out'

cssmuadm@lnx:~$ cat out.txt
working...
working...
working...
working...
working...
working...
cssmuadm@lnx:~$ cat out.txt
working...
working...
working...
working...
working...
working...
working...
cssmuadm@lnx:~$
```

cssmuadm@lnx: ~

```
cssmuadm@lnx:~$ ps aux | grep program
cssmuadm 18959  0.0  0.0   9576  2452 ?        S    14:46
in/bash ./program.sh
cssmuadm 19045  0.0  0.0  11284   940 pts/2    S+   14:48
ep --color=auto program
cssmuadm@lnx:~$ kill -9 18959
cssmuadm@lnx:~$
```

Exercise

Run **sig** program in background detached from terminal (using nohup and &). Close terminal, connect to the server again, check if it still works and kill it

named pipes

mkfifo command creates a named pipe. Works like a regular pipe: in the example below cat will wait for data from pipe until output of the ps command is in the npipe

cssmuadm@lnx: ~/t

```
cssmuadm@lnx:~/t$ mkfifo npipe
cssmuadm@lnx:~/t$ ps > npipe
cssmuadm@lnx:~/t$
```

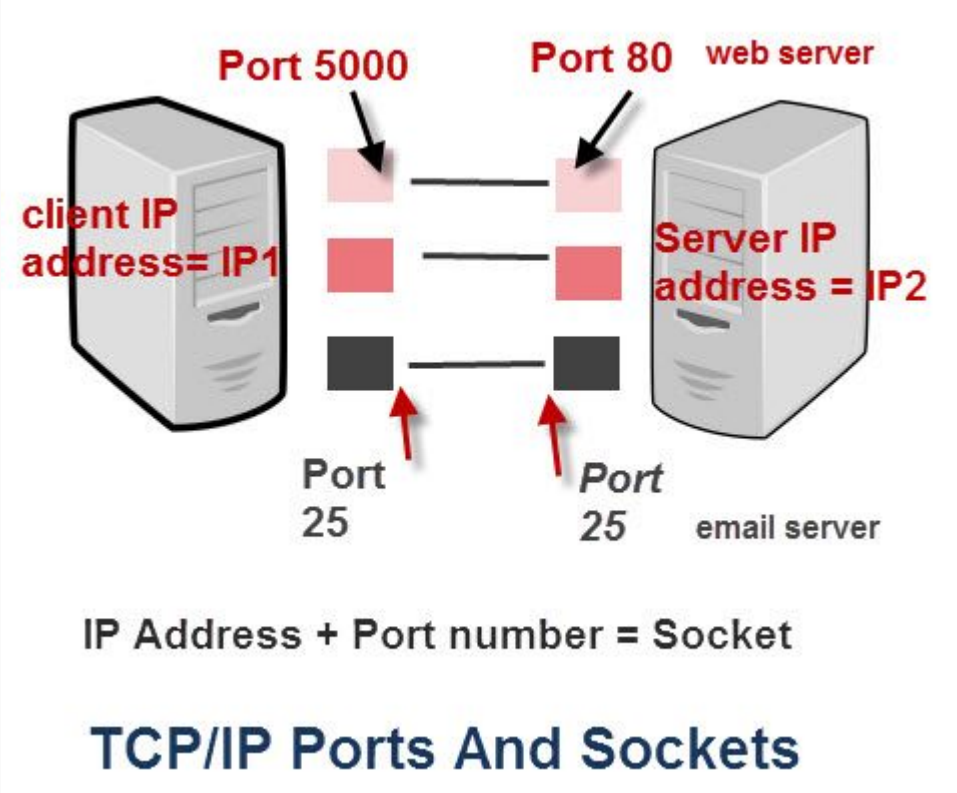
cssmuadm@lnx: ~/t

```
cssmuadm@lnx:~/t$ ls -l npipe
prw-rw-r-- 1 cssmuadm cssmuadm 0 May 18 16:39 npipe
cssmuadm@lnx:~/t$ cat npipe
  PID TTY          TIME CMD
19103 pts/0    00:00:00 bash
19428 pts/0    00:00:00 ps
cssmuadm@lnx:~/t$
```

sockets

socket is a universal communication method between two processes or applications. Applications can run on different machines.

- Sockets use client-server model
- Server **listens** on some port and ip address
- Client **connects** to server
- If server application listens on localhost (127.0.0.1) then only clients running on the same machine can connect to the server application (example: MySQL)
- There is also unix domain sockets that are used to communicate between processes on the same machine (network is not used)



netstat

netstat command can be used to list opened sockets

- -a is used to list both listening and non-listening sockets
- --numeric-ports is used to display port numbers instead of service names

cssmuadm@lnx: ~/.ssh

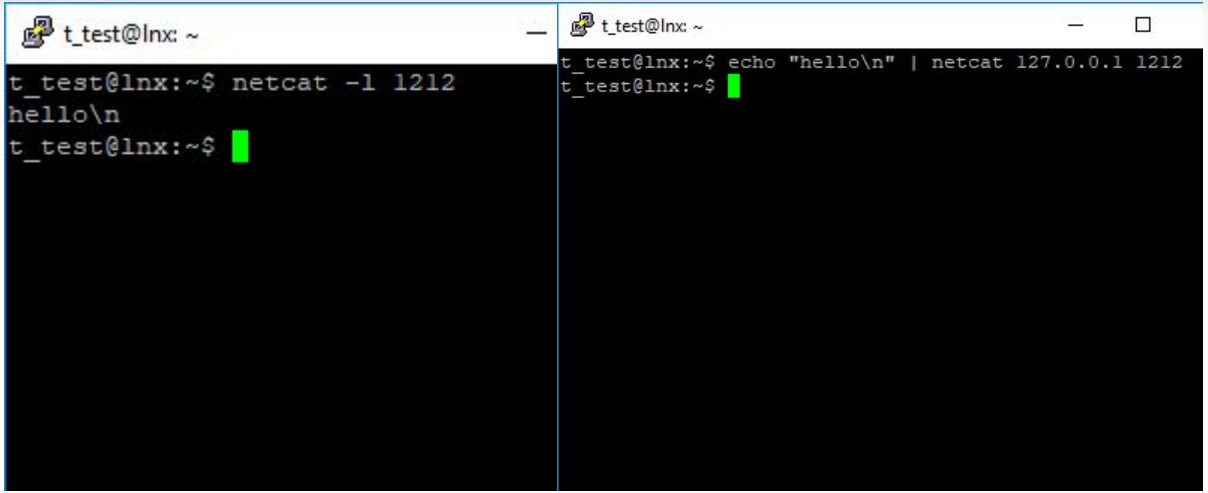
```
cssmuadm@lnx:~/.ssh$ netstat -a --numeric-ports
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0    256 140.184.230.220:22      140.184.193.137:50990   ESTABLISHED
tcp6       0      0 127.0.0.1:8005          :::*                    LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::8080                 :::*                    LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      1 140.184.230.220:80      104.223.203.234:47615   FIN_WAIT1
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     10623    /run/systemd/private
unix  2      [ ]         DGRAM                    845772   /run/user/1000/systemd
/notify
unix  2      [ ACC ]     STREAM     LISTENING     845773   /run/user/1000/systemd
/private
unix  2      [ ACC ]     SEQPACKET  LISTENING     8745     /run/udev/control
unix  2      [ ACC ]     STREAM     LISTENING     10632    /run/systemd/journal/s
tdout
unix  7      [ ]         DGRAM                    10633    /run/systemd/journal/s
ocket
unix  8      [ ]         DGRAM                    8746     /run/systemd/journal/d
ev-log
unix  2      [ ]         DGRAM                    8747     /run/systemd/journal/s
yslog
```

netcat

nc (or netcat)

Can open TCP connections, send packets, listen on arbitrary TCP and UDP ports, do port scanning

Great tool for testing purposes



Exercise

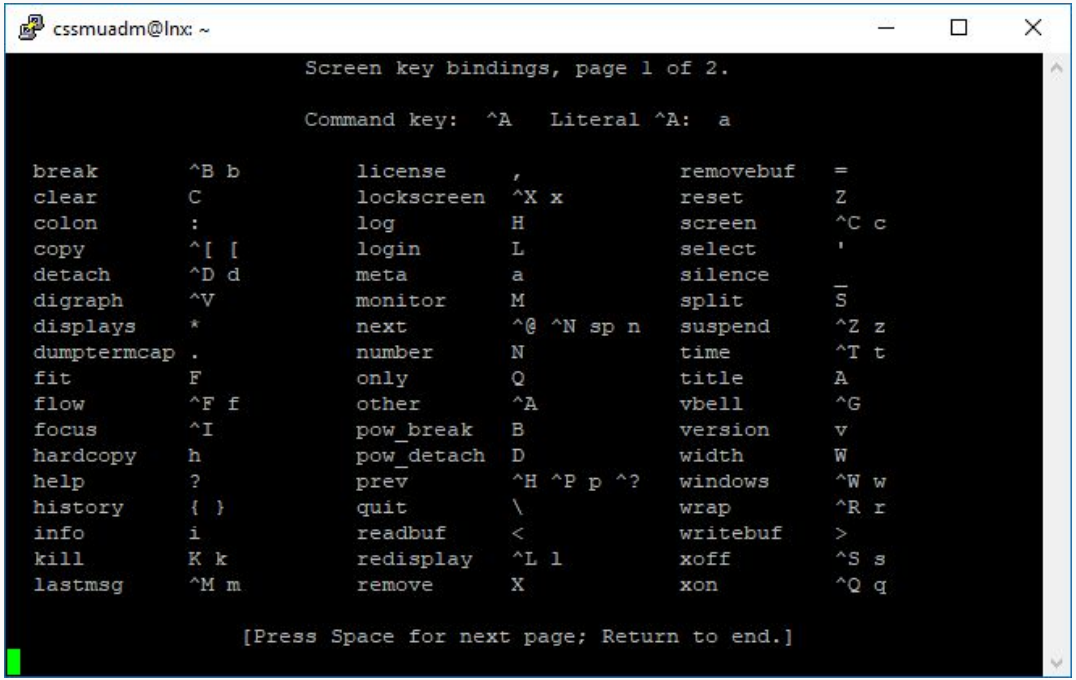
1) Listen on TCP port = 1MMDD where **MM** is you month of birth, **DD** is your day of birth using netcat (with -v flag for verbose output).
2) In other terminal send some text to the socket. What the source port of the connection (**sport** - should appear in the first terminal)?

screen

screen command is used to switch between terminal “windows” inside one terminal.

Start screen and continue to work

- **Ctrl+A c** sequence creates “new screen”
- **Ctrl+A k** sequence kills the current screen
- **Ctrl+A n** sequence goes to the next screen
- **Ctrl+A 0** sequence goes to the first screen, **Ctrl+A 1** goes to the second screen, etc.
- **Ctrl+A ** sequence is used to exit screen

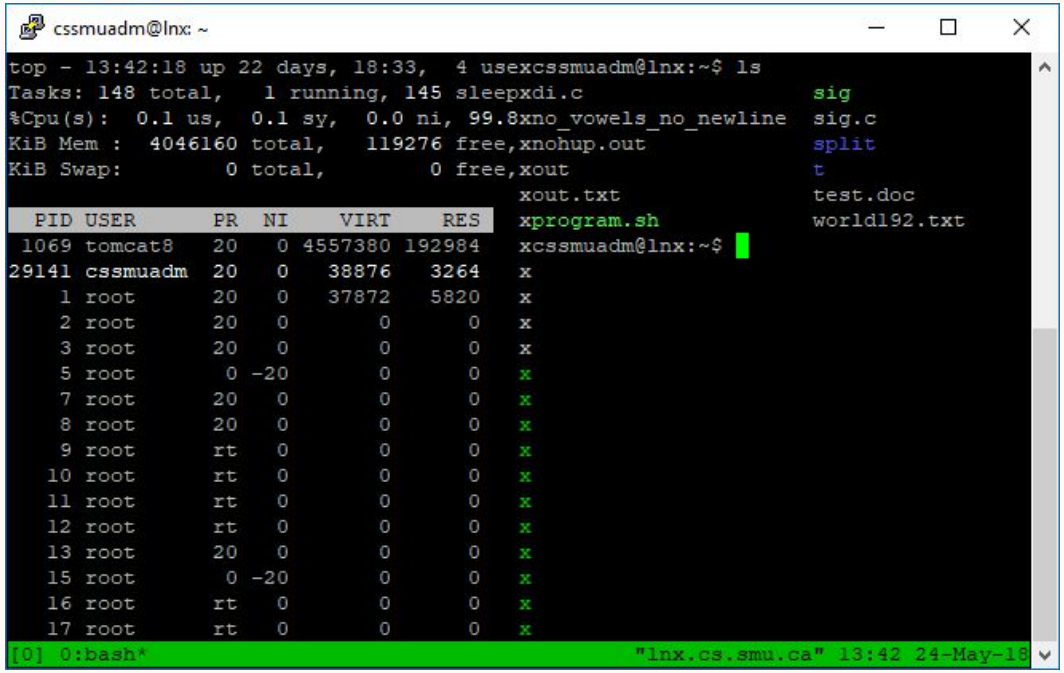


Ctrl+A d detaches from the screen. Even if you close the terminal, the processes attached to the screen will continue to work. Use screen -r and select session to attach to the screen again

tmux

tmux is another “terminal multiplexer” but has better interface and is easier to use than screen

- **Ctrl+B c** sequence creates “new screen”
- **Ctrl+B x** sequence kills the current screen
- **Ctrl+B n** sequence goes to the next screen
- **Ctrl+B 0** sequence goes to the first screen, **Ctrl+A 1** goes to the second screen, etc.
- **Ctrl+B d** sequence is used to detach from screen
- **Ctrl+B %** sequence is used to split sessions vertically, Ctrl+B “ - horizontally
- **tmux attach** attaches to the existing sessions even after terminal window is closed



Exercise

Run **tmux**, in tmux session run **ps**, detach from the screen and close terminal window, reconnect to the server and attach to the existing session with ps running