Computer System & Network Administration

Lecture 05. Process Management

refer to: https://nasa.cs.nctu.edu.tw/sa/2020/

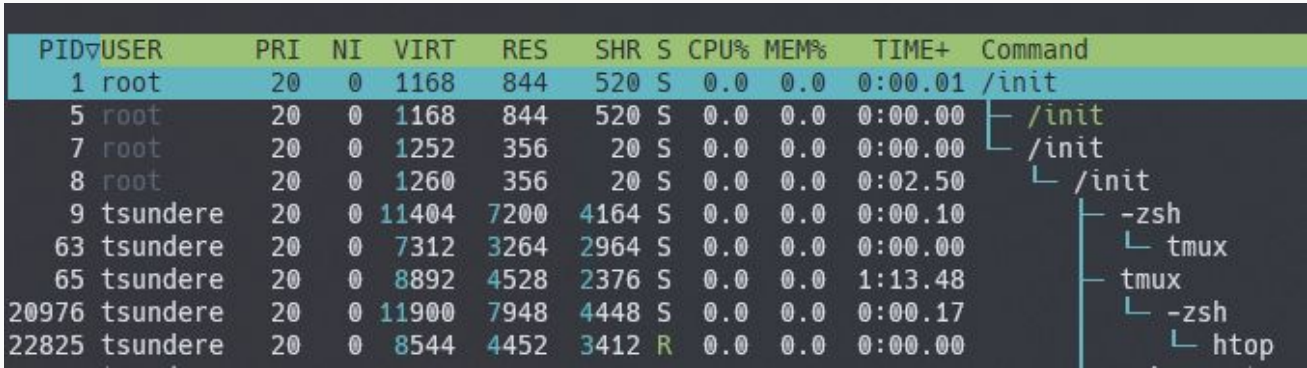
Outline

- Controlling Process
  - Fork Bomb
- Periodic Processes
- Process Resource Management

Controlling Process

From Program to Process

- Program is dead
  - It's just a binary file in your system
  - "executable file" is a program
- When you execute it
  - It becomes a process
- Process is alive
  - Resizes in memory



Components of a Process

- An address space in memory
  - Code and data of this process
- A set of data structures within the kernel
  - Being used to monitor, schedule, trace...this process
    - Owner information
    - Current status
    - VM space
    - Execution priority
    - Information of used resource
    - Resource limits
    - Syscall vector
    - Signal actions

Attributes of the Process

- PID, PPID
  - Process ID and parent process ID
- UID, EUID
  - User ID and Effective user ID
- GID, EGID
  - Group ID and Effective group ID
- Niceness
  - The suggested priority of this process

Attributes of the Process - PID and PPID

- PID - Process ID
  - Unique number assigned for each process in increasing order when they are created
- PPID - parent PID
  - The PID of the parent from which it was cloned
  - UNIX uses fork-and-exec model to create new process

Attributes of the Processes - PID and PPID

pid_ppid.c

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

int main(void){
    int pid;
    pid = fork();
    if(pid == 0){
        // Child Process
        for(int i = 0; i < 12; i++){
            printf("I'm child process, my PID is %d, parent PID is %d\n", getpid(), getppid());
            sleep(1);
        }
        exit(1);
    }
    else if(pid > 0){
        // Parent Process
        for(int i = 0; i < 12; i++){
            printf("I'm parent process, my PID is %d, parent PID is %d\n", getpid(), getppid());
            sleep(1);
        }
    }
    else {
        printf("Fork Error!\n");
        return 0;
    }
}
```

Source

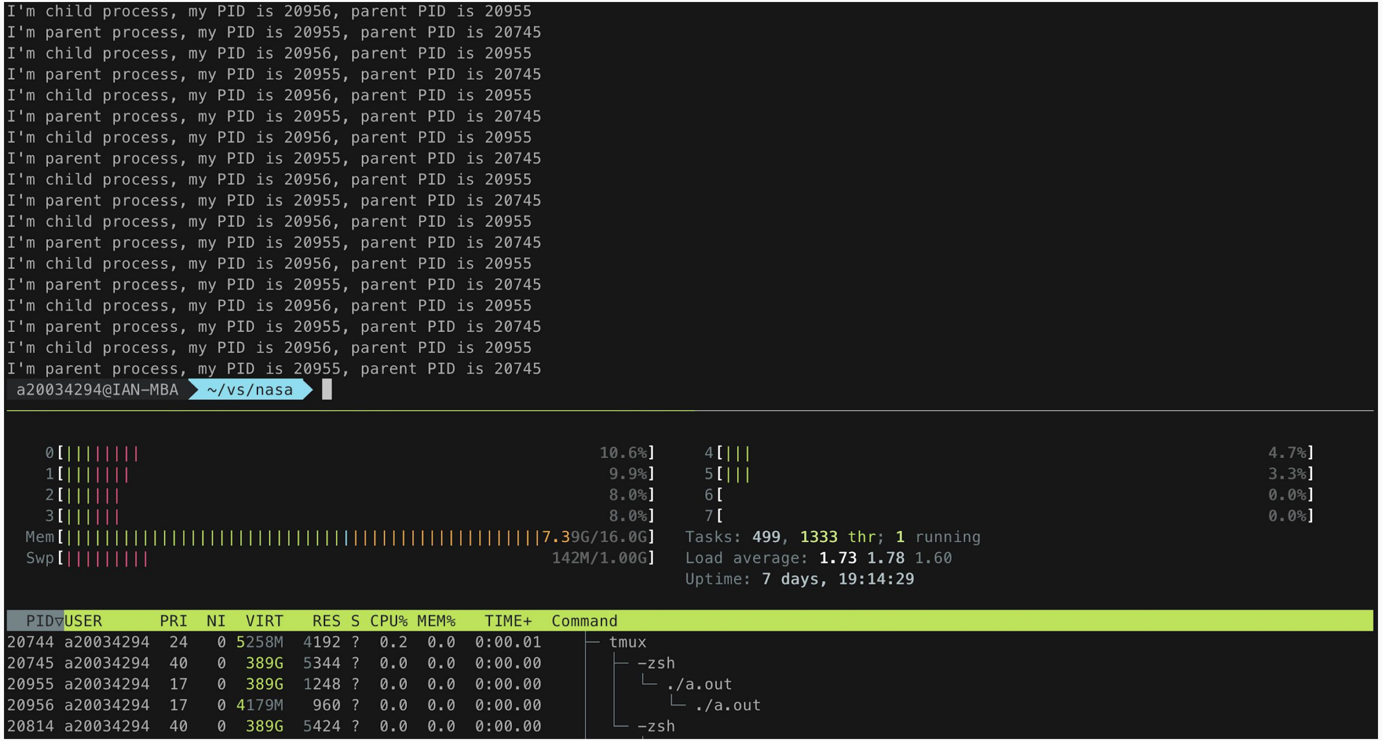
Attributes of the Process - PID and PPID

Process 20955's PPID is 20745

Process 20956's PPID is 20955

Parent PID (relative)

PID



Source

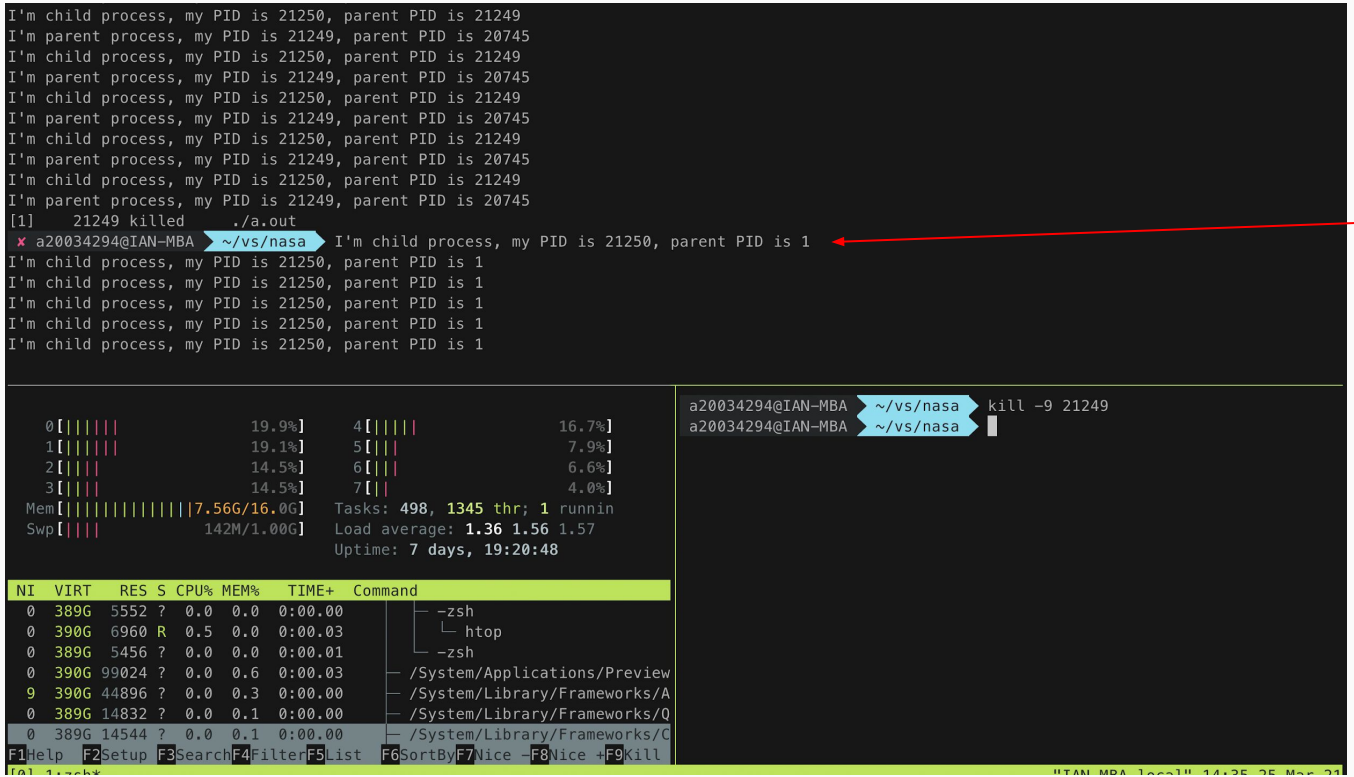
Process Lifecycle

- fork
  - Child has the same program context - fork
- exec
  - Child use exec to change the program context - execve
- exit
  - Child use _exit to tell kernel that it's ready to die and this death should be acknowledged by the child's parent - _exit
- wait
  - Parent use wait to wait for child's death
  - If parent died before child, this orphan process will have init as it's new parent - wait

Process Lifecycle - fork vs. exec

- fork
  - Makes a duplicate of the current process, identical in almost every way
  - The new process gets a different process ID and has the PID of the old process as it's parent PID
- exec
  - Replace the entire current process with a new program
  - Loads the program into the current process space and runs it from the entry point

Process Lifecycle - wait - kill



Notice the change of PPID

Attributes of the Process - UID/GID/EUID/EGID

- UID, GID, EUID, EGID
  - The effective UID/GID can be used to enable or restrict the additional permissions
  - EUID will be set to
    - Real UID if setuid bit is off
    - The file owner's UID if setuid bit is on
  - Example
    - $ ll /usr/bin/chsh
      -rwsr-xr-x 1 root root 31K Feb 12 22:58 /usr/bin/chsh

Signal

- A way of telling a process something has happened
- Signals can be sent
  - Among processes as a means of communication
  - By the terminal driver to kill / interrupt / suspend process
    - <Ctrl-C>, <Ctrl-Z>
    - bg, fg
  - By the administrator to achieve various results
    - With kill
  - By the kernel when a process violate the rules
    - Divide by zero
    - Illegal memory access

Signal - Actions when receiving signal

- Depend on whether there's a designated handler routine for what signal
  - If yes, the handler is called
  - If no, the kernel takes some default action
- "Catching" the signal
  - Specify a handler routine for a signal within a program
- Two ways to prevent signals from arriving
  - Ignore
    - Just discard it and there's no effect to process
  - Block
    - Queue for delivery until unblocked
    - The handler for a newly unblocked signal is called only once

Signal - Linux signals

- man signal
- include/uapi/asm-generic/signal.h

Signal - Signal Dispositions

- Terminate
  - Default action is to terminate the process
- Ignore
  - Default action is to ignore the signal
- Core Dump
  - Default action is to terminate the process and dump core
- Stop
  - Default action is to stop the process
- Continue
  - Default action is to continue the process if it is currently stopped

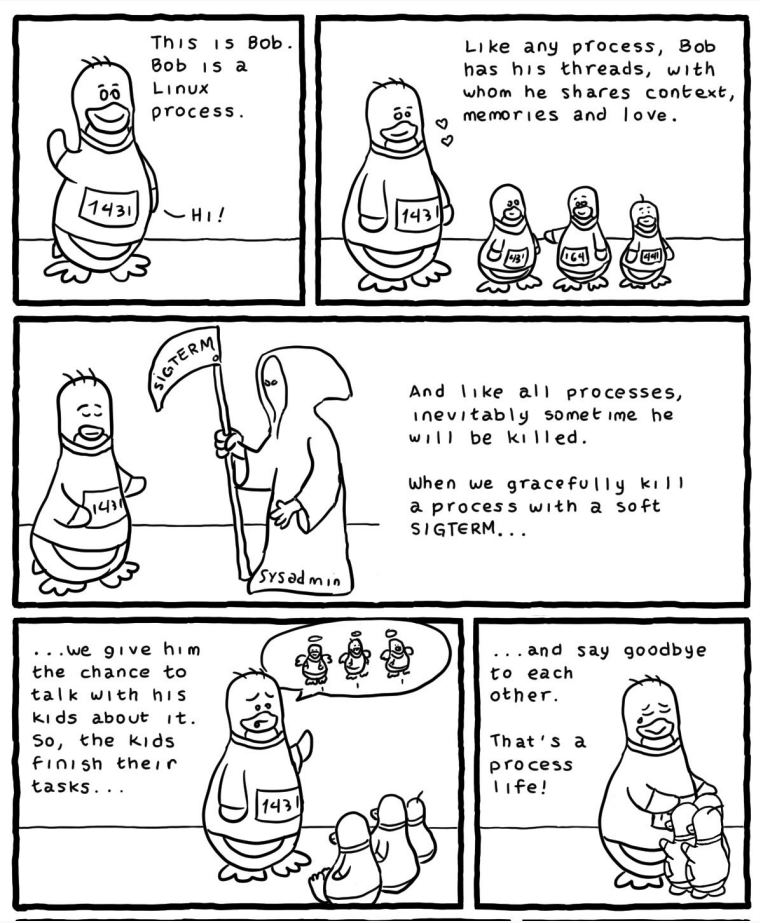
Signal - Linux signals

#	Name	Description	Default	Catch	Block	Core Dump
1	SIGHUP	Hangup	Terminate	Y	Y	N
2	SIGINT	Interrupt (^C)	Terminate	Y	Y	N
3	SIGQUIT	Quit	Core Dump	Y	Y	Y
7	SIGBUS	Bus Error	Core Dump	Y	Y	Y
9	SIGKILL	Kill	Terminate	N	N	N
11	SIGSEGV	Segmentation Fault	Core Dump	Y	Y	Y
15	SIGTERM	Soft Termination	Terminate	Y	Y	N
18	SIGCONT	Continue after stop	Continue	Y	N	N
19	SIGSTOP	Stop	Stop	N	N	N
20	SIGTSTP	Stop from tty (^Z)	Stop	Y	Y	N

Signal - Kill

- man kill
- % kill [-signal] pid
  - First, find out the pid you want to kill
    - ps / top / sockstat / lsof…
  - $ kill -l (List all available signals)
  - $ kill 12345
  - $ kill -TERM 12345
  - $ kill -15 12345
- man killall
  - Kill processes by name
  - $ killall bash
  - $ killall -u ncku-nasa

Signal - Kill - SIGTERM vs. SIGKILL



http://turnoff.us/geek/dont-sigkill/

Niceness

- How kindly of you when contending CPU time
  - High nice value -> Low priority
  - Related to CPU time quantum
- Interent Property
  - A newly created process inherits the nice value of its parent
    - Prevent process with low priority from bearing high-priority children
- Root has complete freedom in setting nice value
  - Use “nice” to set a high-priority shell to beat berserk process

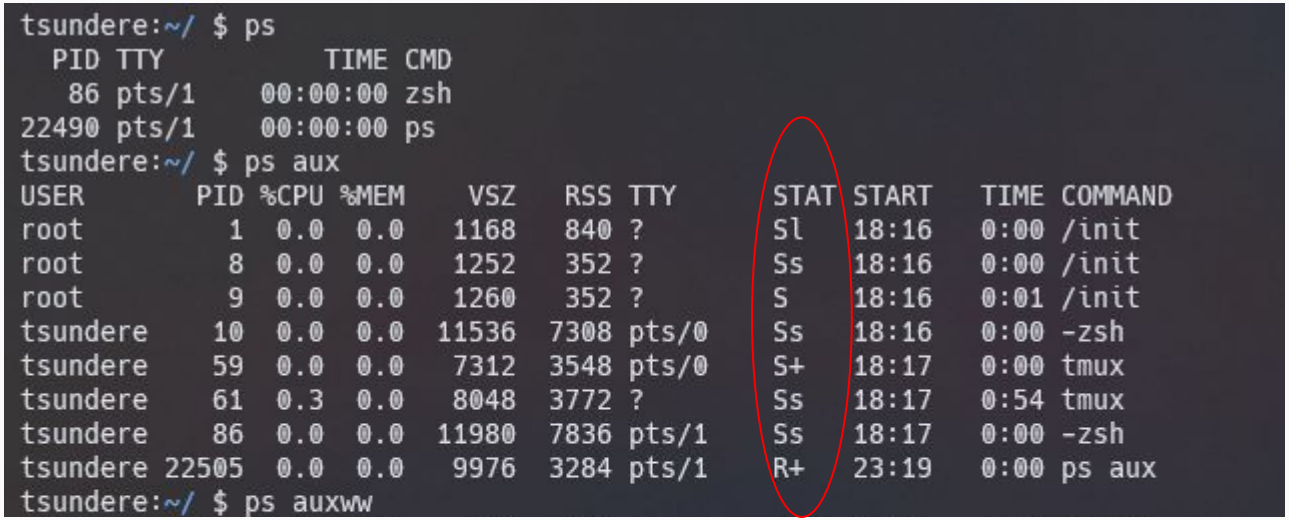
Niceness - nice and renice

- man nice
  - $ nice -n [RANGE] command
- man renice
  - $ renice [RANGE] PID
- Linux nice range
  - root: -20 (Highest priority) ~ +19 (Lowest priority)
  - Normal user: 0 ~ +19
  - Normal user can only set a process' nice higher, not lower

Process States

- man ps
  - ps can show process states

State	Meaning
I	Idle (20+ second)
R	Runnable
S	Sleeping (~20 second)
T	Stopped
Z	Zombie
D	In Disk

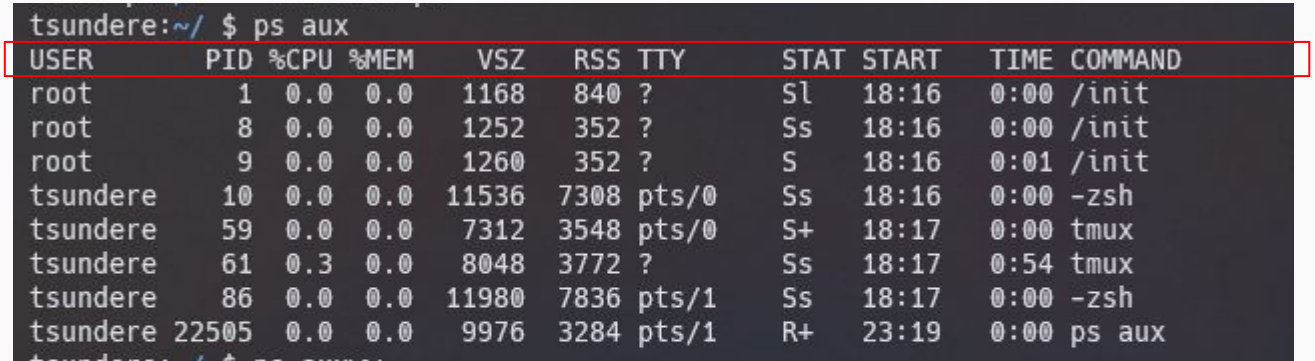


Process States - Zombie Process

- A zombie process is a process that has been terminated, but is still listed as a process (though terminated)
  - It's most likely still listed because it's owned by another process, and that one might request for its status

ps aux - Fields

Field	Description
User	Username of process' owner
PID	Process ID
%CPU	Percentage of CPU this process is using
%MEM	Percentage of real memory this process is using
VSZ	Virtual size of process, in KB
RSS	Resident set size (number of 1K pages in mem)
TTY	Control terminal ID
START	Time the process was started
TIME	CPU time the process has consumed
COMMAND	Command name and arguments



```
tsundere:~/ $ ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.0   1168   840 ?        Sl   18:16   0:00 /init
root         8  0.0  0.0   1252   352 ?        Ss   18:16   0:00 /init
root         9  0.0  0.0   1260   352 ?        S    18:16   0:01 /init
tsundere    10  0.0  0.0  11536  7308 pts/0    Ss   18:16   0:00 -zsh
tsundere    59  0.0  0.0   7312  3548 pts/0    S+   18:17   0:00 tmux
tsundere    61  0.3  0.0   8048  3772 ?        Ss   18:17   0:54 tmux
tsundere    86  0.0  0.0  11980  7836 pts/1    Ss   18:17   0:00 -zsh
tsundere 22505  0.0  0.0   9976  3284 pts/1    R+   23:19   0:00 ps aux
```

ps aux - STAT Field

<table>
<tr><td rowspan="3">STAT</td><td colspan="2">Current process status</td></tr>
<tr><td>

- R: Runnable
- I: Sleeping (> 20 sec)
- S: Sleeping (< 20 sec)

</td><td>

- T: Stopped
- D: In disk (or short-term) wait
- Z: Zombie

</td></tr>
<tr><td colspan="2">

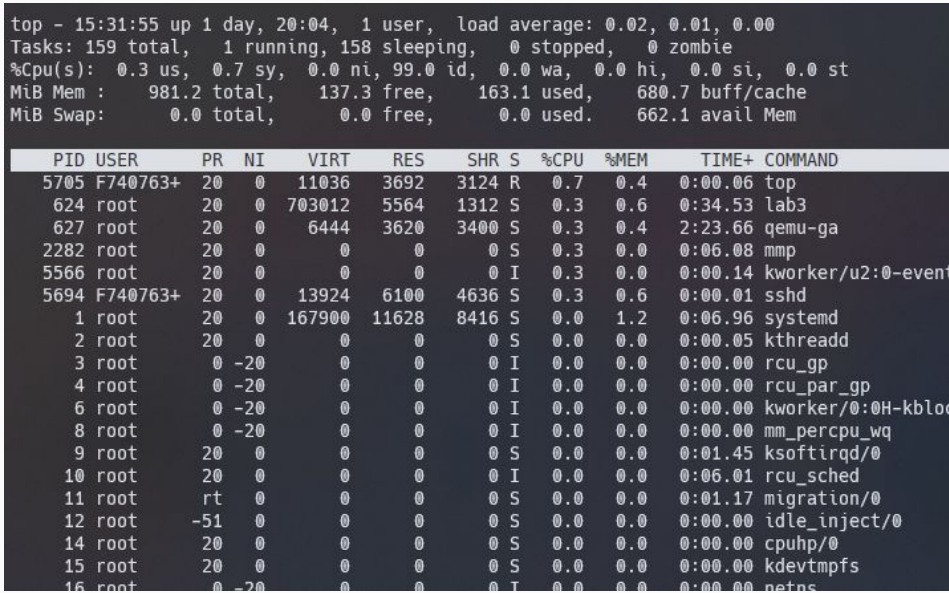
Additional Flags:

- >: Process has higher than normal priority
- N: Process has lower than normal priority
- <: Process is exceeding soft limit on memory usage
- A: Process has requested random page replacement
- S: Process has asked for FIFO page replacement
- V: Process is suspended during a vfork
- E: Process is trying to exit
- L: Some pages are locked in core
- X: Process is being traced or debugged
- s: Process is a session leader (head of controller terminal)
- W: Process is swapped out
- +: Process is in the foreground of its control terminal

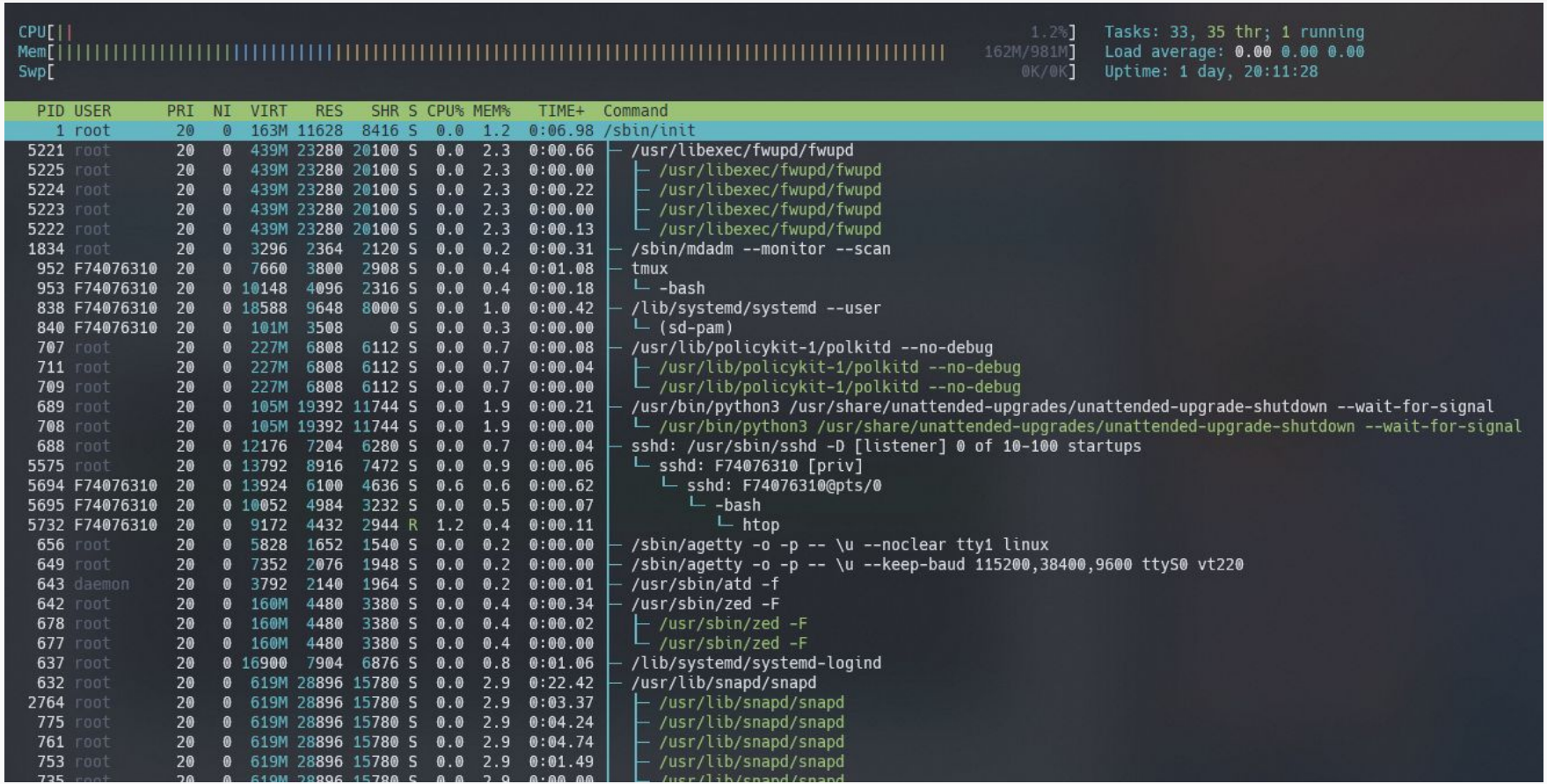
</td></tr>
</table>

top

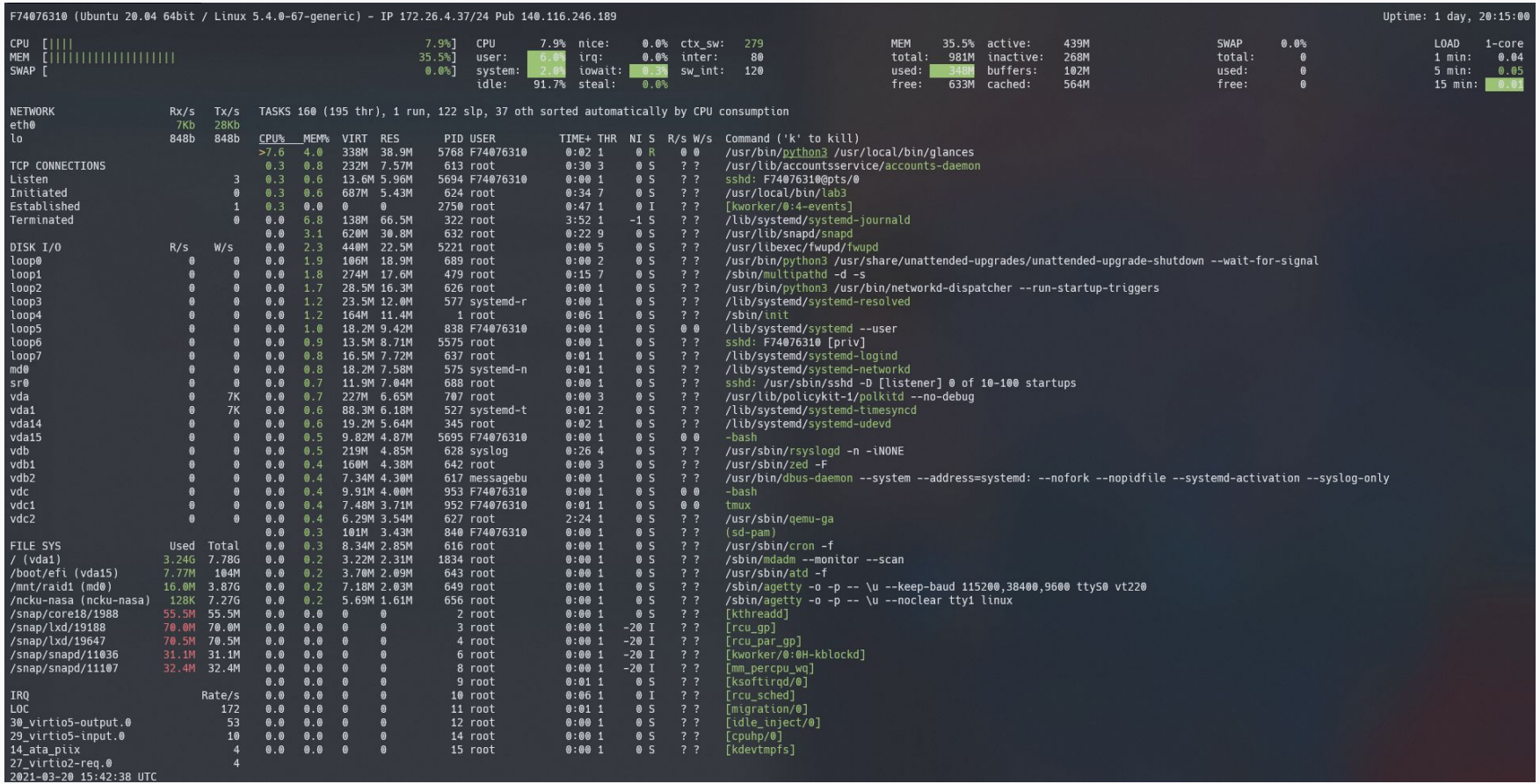
- Various usage
  - top -d <SECOND>
    - Specifies the delay between screen updates
  - top -o <FIELDNAME>
    - Specifies the name of the field on which tasks will be sorted
- Interactive command
  - <Enter> / <Space>
    - Refresh screen
  - H
    - Toggle between showing individual threads
  - r
    - Renice a task



htop


CPU[|
1.2%]
Tasks: 33, 35 thr; 1 running
Mem[
162M/981M]
Load average: 0.00 0.00 0.00
Swp[
0K/0K]
Uptime: 1 day, 20:11:28
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
1 root 20 0 163M 11628 8416 S 0.0 1.2 0:06.98 /sbin/init
5221 root 20 0 439M 23280 20100 S 0.0 2.3 0:00.66 /usr/libexec/fwupd/fwupd
5225 root 20 0 439M 23280 20100 S 0.0 2.3 0:00.00 /usr/libexec/fwupd/fwupd
5224 root 20 0 439M 23280 20100 S 0.0 2.3 0:00.22 /usr/libexec/fwupd/fwupd
5223 root 20 0 439M 23280 20100 S 0.0 2.3 0:00.00 /usr/libexec/fwupd/fwupd
5222 root 20 0 439M 23280 20100 S 0.0 2.3 0:00.13 /usr/libexec/fwupd/fwupd
1834 root 20 0 3296 2364 2120 S 0.0 0.2 0:00.31 /sbin/mdadm --monitor --scan
952 F74076310 20 0 7660 3800 2908 S 0.0 0.4 0:01.08 tmux
953 F74076310 20 0 10148 4096 2316 S 0.0 0.4 0:00.18 -bash
838 F74076310 20 0 18588 9648 8000 S 0.0 1.0 0:00.42 /lib/systemd/systemd --user
840 F74076310 20 0 101M 3508 0 S 0.0 0.3 0:00.00 (sd-pam)
707 root 20 0 227M 6808 6112 S 0.0 0.7 0:00.08 /usr/lib/policykit-1/polkitd --no-debug
711 root 20 0 227M 6808 6112 S 0.0 0.7 0:00.04 /usr/lib/policykit-1/polkitd --no-debug
709 root 20 0 227M 6808 6112 S 0.0 0.7 0:00.00 /usr/lib/policykit-1/polkitd --no-debug
689 root 20 0 105M 19392 11744 S 0.0 1.9 0:00.21 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
708 root 20 0 105M 19392 11744 S 0.0 1.9 0:00.00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
688 root 20 0 12176 7204 6280 S 0.0 0.7 0:00.04 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
5575 root 20 0 13792 8916 7472 S 0.0 0.9 0:00.06 sshd: F74076310 [priv]
5694 F74076310 20 0 13924 6100 4636 S 0.6 0.6 0:00.62 sshd: F74076310@pts/0
5695 F74076310 20 0 10052 4984 3232 S 0.0 0.5 0:00.07 -bash
5732 F74076310 20 0 9172 4432 2944 R 1.2 0.4 0:00.11 htop
656 root 20 0 5828 1652 1540 S 0.0 0.2 0:00.00 /sbin/agetty -o -p -- \u --noclear tty1 linux
649 root 20 0 7352 2076 1948 S 0.0 0.2 0:00.00 /sbin/agetty -o -p -- \u --keep-baud 115200,38400,9600 ttyS0 vt220
643 daemon 20 0 3792 2140 1964 S 0.0 0.2 0:00.01 /usr/sbin/atd -f
642 root 20 0 160M 4480 3380 S 0.0 0.4 0:00.34 /usr/sbin/zed -F
678 root 20 0 160M 4480 3380 S 0.0 0.4 0:00.02 /usr/sbin/zed -F
677 root 20 0 160M 4480 3380 S 0.0 0.4 0:00.00 /usr/sbin/zed -F
637 root 20 0 16900 7904 6876 S 0.0 0.8 0:01.06 /lib/systemd/systemd-logind
632 root 20 0 619M 28896 15780 S 0.0 2.9 0:22.42 /usr/lib/snapd/snapd
2764 root 20 0 619M 28896 15780 S 0.0 2.9 0:03.37 /usr/lib/snapd/snapd
775 root 20 0 619M 28896 15780 S 0.0 2.9 0:04.24 /usr/lib/snapd/snapd
761 root 20 0 619M 28896 15780 S 0.0 2.9 0:04.74 /usr/lib/snapd/snapd
753 root 20 0 619M 28896 15780 S 0.0 2.9 0:01.49 /usr/lib/snapd/snapd

glances


F74076310 (Ubuntu 20.04 64bit / Linux 5.4.0-67-generic) - IP 172.26.4.37/24 Pub 140.116.246.189
Uptime: 1 day, 20:15:00
CPU [|||| 7.9%]
MEM [|||||||||||||||||||| 35.5%]
SWAP [0.0%]
CPU 7.9% nice: 0.0% ctx_sw: 279
user: 6.0% irq: 0.0% inter: 80
system: 2.0% iowait: 0.3% sw_int: 120
idle: 91.7% steal: 0.0%
MEM 35.5% active: 439M
total: 981M inactive: 268M
used: 348M buffers: 102M
free: 633M cached: 564M
SWAP 0.0%
total: 0
used: 0
free: 0
LOAD 1-core
1 min: 0.04
5 min: 0.05
15 min: 0.01
NETWORK Rx/s Tx/s
eth0 7Kb 28Kb
lo 848b 848b
TCP CONNECTIONS
Listen 3
Initiated 0
Established 1
Terminated 0
DISK I/O R/s W/s
loop0 0 0
loop1 0 0
loop2 0 0
loop3 0 0
loop4 0 0
loop5 0 0
loop6 0 0
loop7 0 0
md0 0 0
sr0 0 0
vda 0 7K
vda1 0 7K
vda14 0 0
vda15 0 0
vdb 0 0
vdb1 0 0
vdb2 0 0
vdc 0 0
vdc1 0 0
vdc2 0 0
FILE SYS Used Total
/ (vda1) 3.24G 7.78G
/boot/efi (vda15) 7.77M 104M
/mnt/raid1 (md0) 16.0M 3.87G
/ncku-nasa (ncku-nasa) 128K 7.27G
/snap/core18/1988 55.5M 55.5M
/snap/lxd/19188 70.0M 70.0M
/snap/lxd/19647 70.5M 70.5M
/snap/snapd/11036 31.1M 31.1M
/snap/snapd/11107 32.4M 32.4M
IRQ Rate/s
LOC 172
30_virtio5-output.0 53
29_virtio5-input.0 10
14_ata_piix 4
27_virtio2-req.0 4
2021-03-20 15:42:38 UTC
TASKS 160 (195 thr), 1 run, 122 slp, 37 oth sorted automatically by CPU consumption
CPU% MEM% VIRT RES PID USER TIME+ THR NI S R/s W/s Command ('k' to kill)
>7.6 4.0 338M 38.9M 5768 F74076310 0:02 1 0 R 0 0 /usr/bin/python3 /usr/local/bin/glances
0.3 0.8 232M 7.57M 613 root 0:30 3 0 S ? ? /usr/lib/accountsservice/accounts-daemon
0.3 0.6 13.6M 5.96M 5694 F74076310 0:00 1 0 S ? ? sshd: F74076310@pts/0
0.3 0.6 687M 5.43M 624 root 0:34 7 0 S ? ? /usr/local/bin/lab3
0.3 0.0 0 0 2750 root 0:47 1 0 I ? ? [kworker/0:4-events]
0.0 6.8 138M 66.5M 322 root 3:52 1 -1 S ? ? /lib/systemd/systemd-journald
0.0 3.1 620M 30.8M 632 root 0:22 9 0 S ? ? /usr/lib/snapd/snapd
0.0 2.3 440M 22.5M 5221 root 0:00 5 0 S ? ? /usr/libexec/fwupd/fwupd
0.0 1.9 106M 18.9M 689 root 0:00 2 0 S ? ? /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
0.0 1.8 274M 17.6M 479 root 0:15 7 0 S ? ? /sbin/multipathd -d -s
0.0 1.7 28.5M 16.3M 626 root 0:00 1 0 S ? ? /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup-triggers
0.0 1.2 23.5M 12.0M 577 systemd-r 0:00 1 0 S ? ? /lib/systemd/systemd-resolved
0.0 1.2 164M 11.4M 1 root 0:06 1 0 S ? ? /sbin/init
0.0 1.0 18.2M 9.42M 838 F74076310 0:00 1 0 S 0 0 /lib/systemd/systemd --user
0.0 0.9 13.5M 8.71M 5575 root 0:00 1 0 S ? ? sshd: F74076310 [priv]
0.0 0.8 16.5M 7.72M 637 root 0:01 1 0 S ? ? /lib/systemd/systemd-logind
0.0 0.8 18.2M 7.58M 575 systemd-n 0:01 1 0 S ? ? /lib/systemd/systemd-networkd
0.0 0.7 11.9M 7.04M 688 root 0:00 1 0 S ? ? sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
0.0 0.7 227M 6.65M 707 root 0:00 3 0 S ? ? /usr/lib/policykit-1/polkitd --no-debug
0.0 0.6 88.3M 6.18M 527 systemd-t 0:01 2 0 S ? ? /lib/systemd/systemd-timesyncd
0.0 0.6 19.2M 5.64M 345 root 0:02 1 0 S ? ? /lib/systemd/systemd-udevd
0.0 0.5 9.82M 4.87M 5695 F74076310 0:00 1 0 S 0 0 -bash
0.0 0.5 219M 4.85M 628 syslog 0:26 4 0 S ? ? /usr/sbin/rsyslogd -n -iNONE
0.0 0.4 160M 4.38M 642 root 0:00 3 0 S ? ? /usr/sbin/zed -F
0.0 0.4 7.34M 4.30M 617 messagebu 0:00 1 0 S ? ? /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
0.0 0.4 9.91M 4.00M 953 F74076310 0:00 1 0 S 0 0 -bash
0.0 0.4 7.48M 3.71M 952 F74076310 0:01 1 0 S 0 0 tmux
0.0 0.4 6.29M 3.54M 627 root 2:24 1 0 S ? ? /usr/sbin/qemu-ga
0.0 0.3 101M 3.43M 840 F74076310 0:00 1 0 S ? ? (sd-pam)
0.0 0.3 8.34M 2.85M 616 root 0:00 1 0 S ? ? /usr/sbin/cron -f
0.0 0.2 3.22M 2.31M 1834 root 0:00 1 0 S ? ? /sbin/mdadm --monitor --scan
0.0 0.2 3.70M 2.09M 643 root 0:00 1 0 S ? ? /usr/sbin/atd -f
0.0 0.2 7.18M 2.03M 649 root 0:00 1 0 S ? ? /sbin/agetty -o -p -- \u --keep-baud 115200,38400,9600 ttyS0 vt220
0.0 0.2 5.69M 1.61M 656 root 0:00 1 0 S ? ? /sbin/agetty -o -p -- \u --noclear tty1 linux
0.0 0.0 0 0 2 root 0:00 1 0 S ? ? [kthreadd]
0.0 0.0 0 0 3 root 0:00 1 -20 I ? ? [rcu_gp]
0.0 0.0 0 0 4 root 0:00 1 -20 I ? ? [rcu_par_gp]
0.0 0.0 0 0 6 root 0:00 1 -20 I ? ? [kworker/0:0H-kblockd]
0.0 0.0 0 0 8 root 0:00 1 -20 I ? ? [mm_percpu_wq]
0.0 0.0 0 0 9 root 0:01 1 0 S ? ? [ksoftirqd/0]
0.0 0.0 0 0 10 root 0:06 1 0 I ? ? [rcu_sched]
0.0 0.0 0 0 11 root 0:01 1 0 S ? ? [migration/0]
0.0 0.0 0 0 12 root 0:00 1 0 S ? ? [idle_inject/0]
0.0 0.0 0 0 14 root 0:00 1 0 S ? ? [cpuhp/0]
0.0 0.0 0 0 15 root 0:00 1 0 S ? ? [kdevtmpfs]

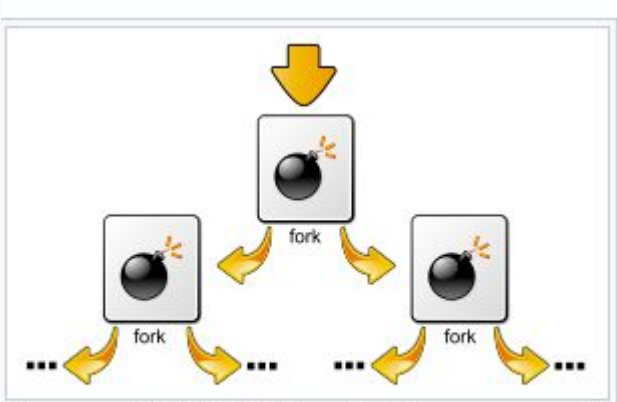
Runaway process

- Processes that use up excessive system resources or just go berserk
  - kill -TERM for unknown process
  - renice it to a higher nice value for reasonable process

Fork Bomb

Fork Bomb

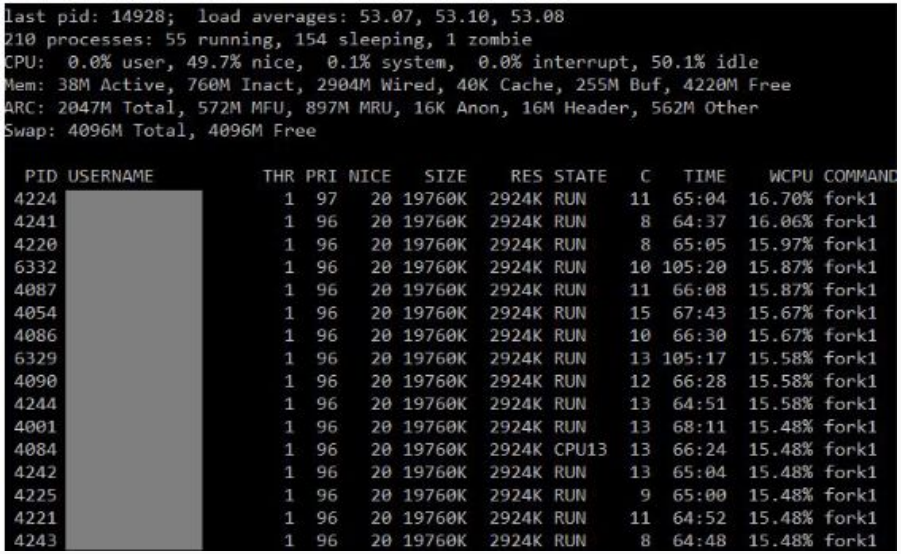
- A process forking out of control



The concept behind a fork bomb — the processes continually replicate themselves, potentially causing a denial of service

Fork Bomb

- A process forking out of control



Fork Bomb - How to create a fork bomb

- C/C++

```c
#include <unistd.h>

int main(void) {
    while(1)
        fork();
    return 0;
}
```

- Perl

```
fork while fork
```

- Windows

```
%0 | %0
```

- Bash (Shell script)

```
:(){ :|:& };:
```

```
# 定義函式
forkbomb() {
    # 使用 PIPE 呼叫兩次並丟到背景執行
    forkbomb|forkbomb &
}
;
執行函式，引爆 fork bomb
forkbomb
```

Please DO NOT DO THAT!!!!!!

Fork Bomb - Defusing / Preventing the bomb

- How to deal with fork bomb
  - Just kill all of them
    - $ killall -KILL <BOMBNAME>
  - When you have no more resource to fork your shell
    - $ exec killall -KILL <BOMBNAME>
    - That shell will become “killall”, and never goes back
  - `killall` isn’t an atomic command
    - More bombs may be created when killing them
    - Run multiple `killall`

Fork Bomb - Defusing / Preventing the bomb

- Prevent fork bomb
  - Limit the maximum number of processes for a specific user
- /etc/security/limits.conf
  - `nproc`

```
#*              soft    core            0
#root           hard    core            100000
#*              hard    rss             10000
#@student       hard    nproc           20
#@faculty       soft    nproc           20
#@faculty       hard    nproc           50
#ftp            hard    nproc           0
#ftp            -       chroot          /ftp
#@student       -       maxlogins       4
```

Periodic Processes

CRON – Schedule Commands

- What we want?
  - Do things at right time automatically
  - e.x. Auto update packages on every SUNDAY at 03:00
- cron daemon
  - The daemon that handles periodic execution
  - cron daemon reads configuration file and executes commands on time

CRON – Schedule Commands

- Configuration file
  - crontab (cron table)
  - Location of system cron configuration file:
    - /etc/crontab
  - Location of user cron configuration file:

System	Cron Dir
Ubuntu	/var/spool/cron/crontabs
FreeBSD	/var/cron/tabs
CentOS	/var/spools/cron

CRON – Schedule Commands

- Configuration file
  - Every user can have at most one crontab file and this file will be named username

```
F74061030@F74061030 ~ sudo ls -al /var/spool/cron/crontabs/
total 16
drwx-wx--T 2 root      crontab 4096 Mar 20 22:45 .
drwxr-xr-x 5 root      root    4096 Feb 23 05:48 ..
-rw------- 1 F74061030 crontab 1090 Mar 20 22:44 F74061030
-rw------- 1 root      crontab 1090 Mar 20 22:45 root
```

CRON – Schedule Commands

- /etc/crontab
  - system-wide crontab
  - auto run files in /etc/cron.(daily|weekly|monthly)
  - can select user to execute

```
✘ F74061030@F74061030 ~ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly
```

CRON – File Format

1. Ignored
   - Blank lines or leading spaces and tabs
2. Comments
   - by leading pound-sign #
3. Environment setting
   - KEY=VALUE
   - Default env variables: LOGNAME, SHELL, PATH, HOME, MAILTO
     - set these vars. on demand

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

CRON – File Format

4. a cron command

```
# m h d mon dow   command
  0 3 * *   7     /home/F74061030/runsomething -abcdefg --lalala
```

Field	Range
minute of the hour	0 ~ 59
hour of the day	0 ~ 23
day of the month	1 ~ 31
month of the year	1 ~ 12
weekday	0 ~ 7 (0, 7 = Sunday)

CRON – File Format

5. matching rules
    - * matches everything
    - Single character matches exactly
    - Dash (-) matches range
    - Comma (,) matches any listed value
    - Slash (/) matches skips of the number's value through the range.

CRON – File Format

```
*   *   *    *    *      Every minute
6   0   */3  *    *      Every three days at 0:06
0   3   *    *    3,5    Every Wed, Fri at 3:00
0   4   1    12   *      Every 1 Dec at 4:00
36  22  *    7-8  *      Every day in range(Jul, Aug) at 22:36
@reboot                  <-- What's that???
```

CRON - File format

- Nonstandard predefined scheduling definitions

Entry	Description	Equivalent to
@yearly / @annually	Run once a year at midnight of 1 Jan.	0 0 1 1 *
@monthly	Run once a month at midnight of the first day of the month	0 0 1 * *
@weekly	Run once a week at midnight on Sunday morning	0 0 * * 0
@daily / @midnight	Run once a day at midnight	0 0 * * *
@hourly	Run once an hour at the beginning of the hour	0 * * * *
@reboot	Run at startup	N/A

CRON – How to edit

- How to edit
  - $ crontab -e [-u user]
- All you want to know
  - $ crontab -h
  - $ man crontab
  - don’t be afraid to see manual

Process Resource Management

cgroups

- Control groups are a feature provided by the Linux kernel to manage, restrict, and audit groups of processes
- More flexible as they can operate on (sub)sets of processes (possibly with different system users)
- Provided through pseudo-filesystem called cgroupfs

cgroups

- cgroups can be accessed with various tools
  - Using directives in systemd unit files to specify limits for services and slices
  - By accessing the cgroups filesystem directly
  - Via tools like cgcreate, cgexec and cgclassify
  - Using the “rules engine daemon” to automatically move certain users/groups/commands to groups
  - Through other software such as Linux Containers virtualization

cgroups

- Practical uses
  - Isolating core workload from background resource needs
    - Web server vs. system processes
    - Time critical work vs. long-term async. jobs
  - Don’t allow one workload to overpower the others

cgroups - mount

```
F74076310@F74076310:~$ mount | grep cgroup
tmpfs on /sys/fs/cgroup type tmpfs (ro,nosuid,nodev,noexec,mode=755)
cgroup2 on /sys/fs/cgroup/unified type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate)
cgroup on /sys/fs/cgroup/systemd type cgroup (rw,nosuid,nodev,noexec,relatime,xattr,name=systemd)
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,net_prio)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/cpu,cpuacct type cgroup (rw,nosuid,nodev,noexec,relatime,cpu,cpuacct)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/rdma type cgroup (rw,nosuid,nodev,noexec,relatime,rdma)
cgroup on /sys/fs/cgroup/memory type cgroup (rw,nosuid,nodev,noexec,relatime,memory)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
```

cgroups - v1 and v2

- cgroups v1
  - Development started at 2006 in Google by Rohit Seth and Paul Menage
  - Initially this project is called process containers, renamed to control groups to avoid confusion
- cgroups v2
  - Originally developed by Tejun Heo in Facebook, merged into Linux Kernel 4.5.0
    - cgroupv2: Linux's new unified control group system, FOSDEM
  - Unified hierarchy

How did this work in cgroupv1?

cgroupv1 has a hierarchy per-resource, for example:

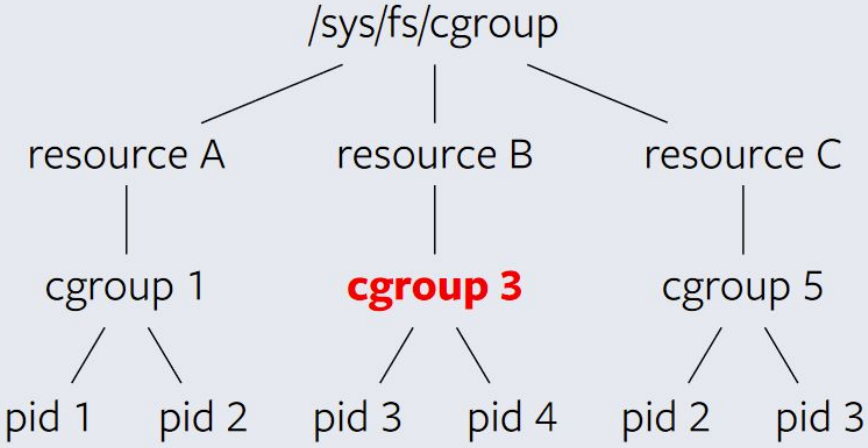
```
% ls /sys/fs/cgroup
cpu/ cpuacct/  cpuset/  devices/  freezer/
memory/  net_cls/  pids/
```

Each resource hierarchy contains cgroups for this resource:

```
% find /sys/fs/cgroup/pids -type d
/sys/fs/cgroup/pids/background.slice
/sys/fs/cgroup/pids/background.slice/async.slice
/sys/fs/cgroup/pids/workload.slice
```

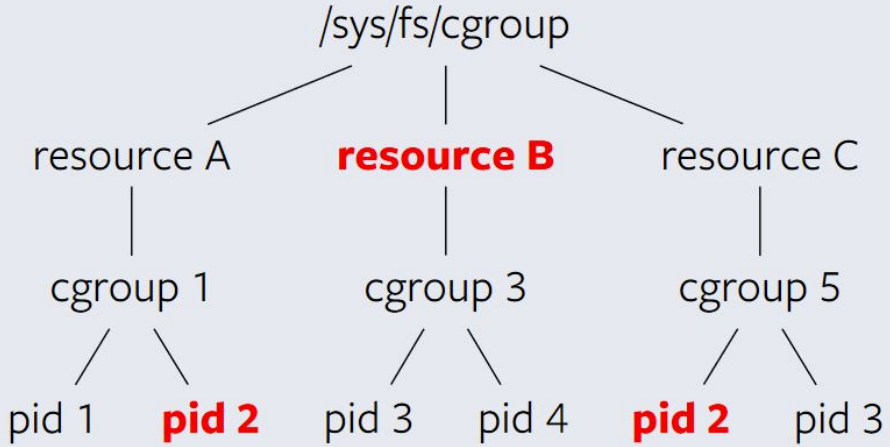
How did this work in cgroupv1?

- Limits and accounting are performed per-cgroup
- If resource B is "memory", you can set `memory.limit_in_bytes` in cgroup 3

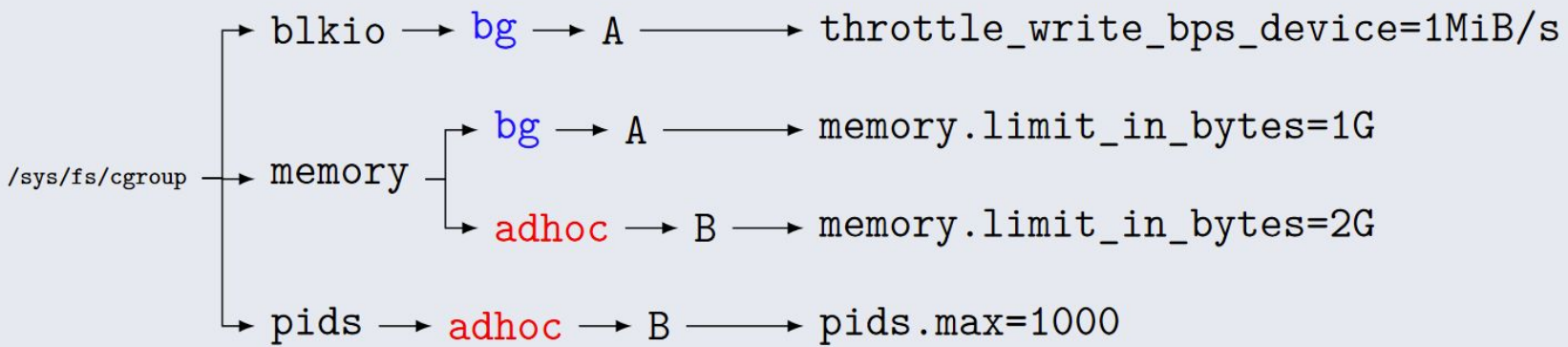


How did this work in cgroupv1?

- One PID is in exactly one cgroup per resource
- PID 2 explicitly assigned in separate cgroups for resource A and C
- Not assigned for resource B, so in the root cgroup



Hierarchy in cgroupv1



How does this work in cgroupv2?

cgroupv2 has a *unified hierarchy*, for example:

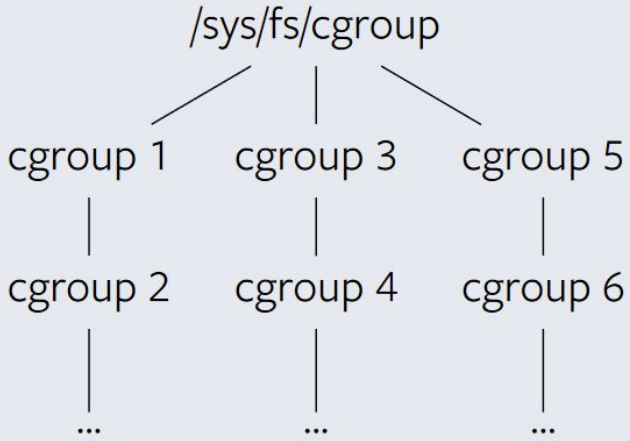
```
% ls /sys/fs/cgroup
background.slice/  workload.slice/
```

Each cgroup can support multiple resource domains:

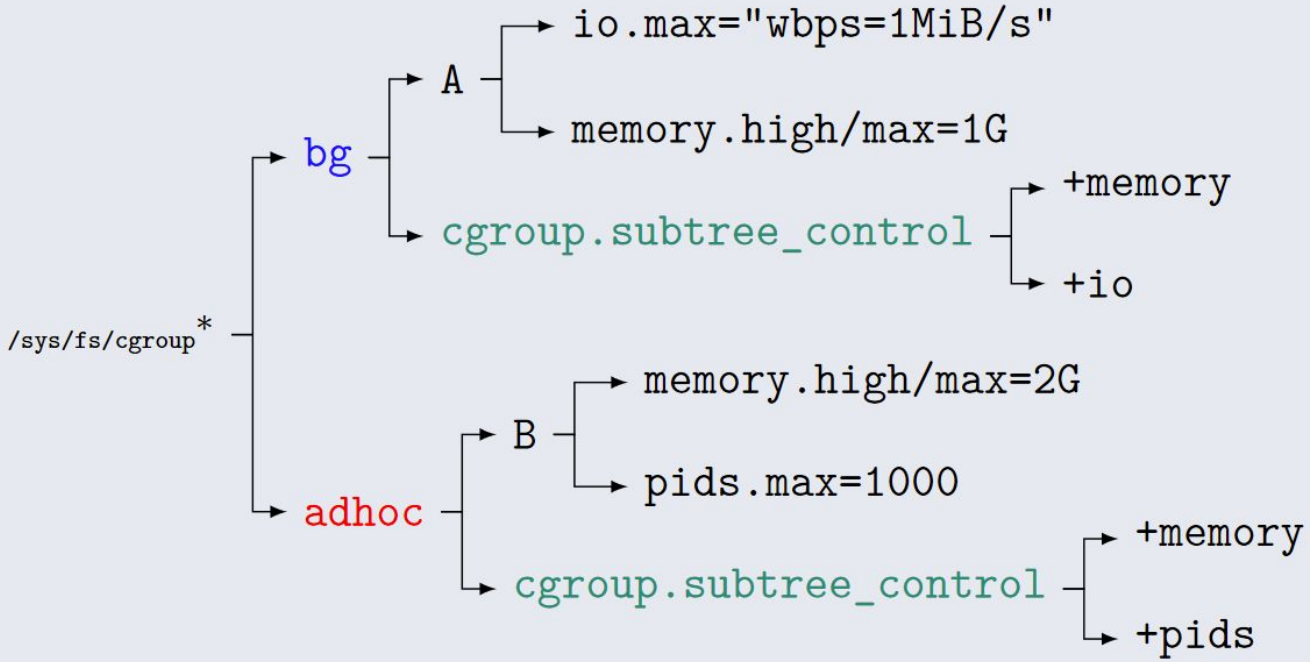
```
% ls /sys/fs/cgroup/background.slice
async.slice/  foo.mount/  cgroup.subtree_control
memory.high  memory.max  pids.current  pids.max
```

How does this work in cgroupv2?

- cgroups are "global" now — not limited to one resource
- Resources are now opt-in for cgroups



Hierarchy in cgroupv2



* in real life, we must enable memory/pids/io controllers for children here too

Fundamental differences between v1 and v2

- Unified hierarchy — resources apply to cgroups now
- Granularity at TGID (PID), not TID level
- Focus on simplicity/clarity over ultimate flexibility

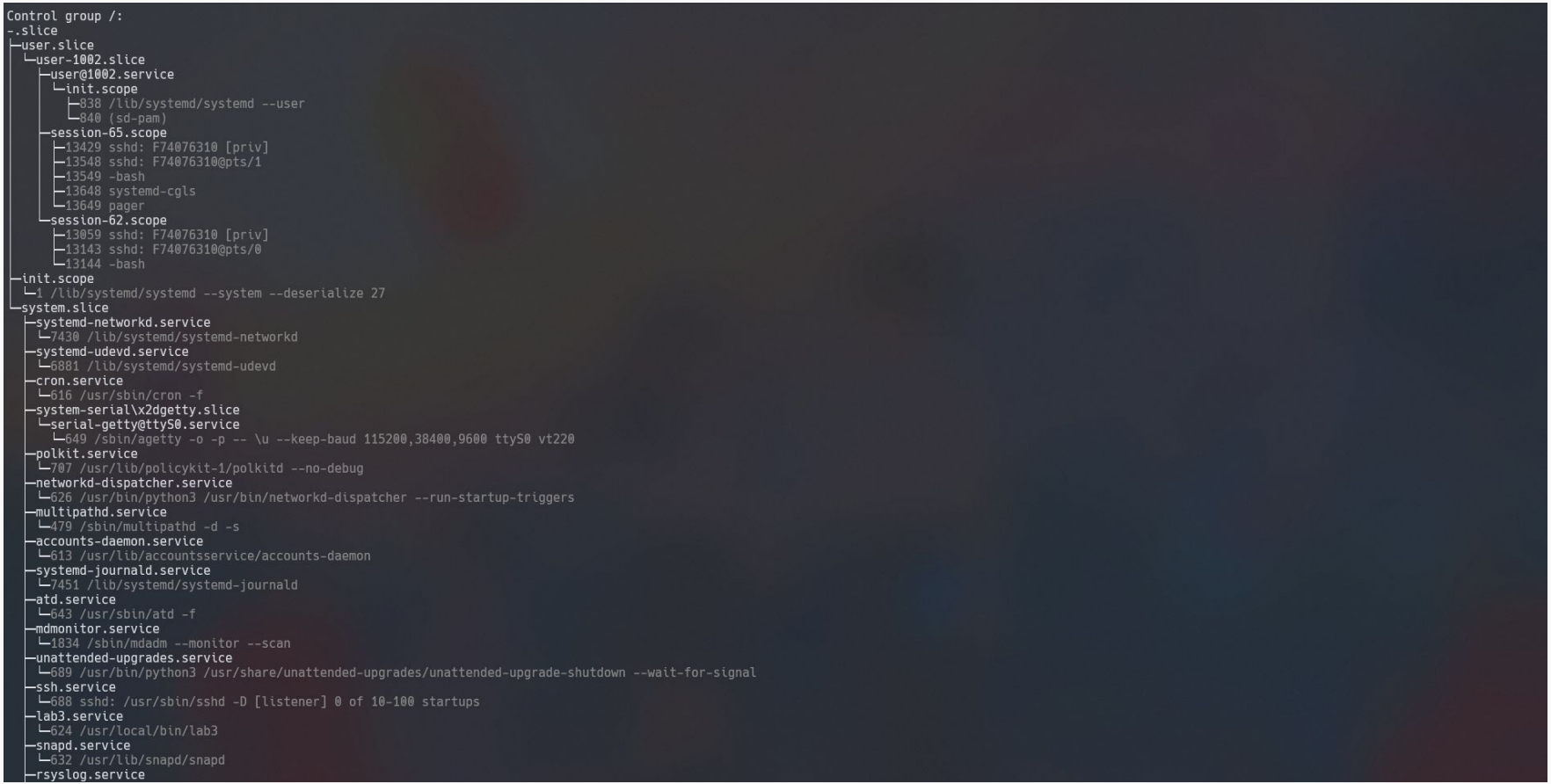
cgroups - visibility

- It can be really complicated if we try to use cgroups in our system directly
  - So the easiest way to do that is with some tools' help, like systemd
- systemd utilizes cgroups v2, but still support cgroups v1
  - Called hybrid set up
- We can use systemd to limit services using cgroups

cgroups - Controller Support

- cgroups v1
  - cpu, cpuacct, blkio, memory, devices, pids, freezer, cpuset, net_cls, perf_event, net_prio, hugetlb
- cgroups v2
  - cpu, io, memory, pids
- Not all of them are supported by systemd

cgroups - systemd hierarchy `systemd-cgls`



cgroups - Find cgroup of a process

$ cat /proc/<PID>/cgroup

```
F74076310@F74076310:~$ cat /proc/self/cgroup
12:devices:/user.slice
11:blkio:/user.slice
10:pids:/user.slice/user-1002.slice/session-62.scope
9:freezer:/
8:memory:/user.slice/user-1002.slice/session-62.scope
7:rdma:/
6:perf_event:/
5:cpu,cpuacct:/user.slice
4:cpuset:/
3:hugetlb:/
2:net_cls,net_prio:/
1:name=systemd:/user.slice/user-1002.slice/session-62.scope
0::/user.slice/user-1002.slice/session-62.scope
```

cgroups - Resource usage `systemd-cgtop`

Control Group	Tasks	%CPU	Memory	Input/s	Output/s
user.slice	6	4.0	279.3M	-	-
/	189	2.9	828.7M	-	-
system.slice	54	0.3	375.9M	-	-
system.slice/systemd-journald.service	1	0.2	4.0M	-	-
system.slice/qemu-guest-agent.service	1	0.1	896.0K	-	-
system.slice/lab3.service	7	0.0	6.0M	-	-
system.slice/accounts-daemon.service	3	0.0	5.5M	-	-
system.slice/rsyslog.service	4	0.0	6.8M	-	-
system.slice/multipathd.service	7	0.0	13.3M	-	-
init.scope	1	-	6.5M	-	-
system.slice/atd.service	1	-	512.0K	-	-
system.slice/boot-efi.mount	-	-	128.0K	-	-
system.slice/cron.service	1	-	3.9M	-	-
system.slice/dbus.service	1	-	2.2M	-	-
system.slice/dev-hugepages.mount	-	-	136.0K	-	-
system.slice/mdmonitor.service	1	-	412.0K	-	-
system.slice/networkd-dispatcher.service	1	-	9.2M	-	-
system.slice/polkit.service	3	-	1.3M	-	-
system.slice/snap-core18-1988.mount	-	-	308.0K	-	-
system.slice/snap-lxd-19188.mount	-	-	116.0K	-	-
system.slice/snap-lxd-19647.mount	-	-	120.0K	-	-
system.slice/snap-snapd-11036.mount	-	-	120.0K	-	-
system.slice/snap-snapd-11107.mount	-	-	124.0K	-	-
system.slice/snapd.service	9	-	42.3M	-	-
system.slice/ssh.service	1	-	5.9M	-	-
system.slice/sys-fs-fuse-connections.mount	-	-	96.0K	-	-
system.slice/sys-kernel-config.mount	-	-	96.0K	-	-
system.slice/sys-kernel-debug.mount	-	-	76.0K	-	-
system.slice/sys-kernel-tracing.mount	-	-	68.0K	-	-
system.slice/system-getty.slice	1	-	308.0K	-	-
system.slice/system-getty.slice/getty@tty1.service	1	-	308.0K	-	-
system.slice/system-modprobe.slice	-	-	64.0K	-	-
system.slice/system-serial\x2dgetty.slice	1	-	292.0K	-	-
system.slice/system-serial\x2dgetty.slice/serial-getty@ttyS0.service	1	-	292.0K	-	-
system.slice/system-systemd\x2dfsck.slice	-	-	48.0K	-	-
system.slice/systemd-logind.service	1	-	2.9M	-	-
system.slice/systemd-networkd.service	1	-	1.3M	-	-
system.slice/systemd-resolved.service	1	-	4.4M	-	-
system.slice/systemd-timesyncd.service	2	-	1.2M	-	-
system.slice/systemd-udevd.service	1	-	1.9M	-	-
system.slice/unattended-upgrades.service	2	-	11.5M	-	-
system.slice/zfs-zed.service	3	-	984.0K	-	-
user.slice/user-1002.slice	6	-	279.2M	-	-
user.slice/user-1002.slice/session-62.scope	4	-	6.8M	-	-
user.slice/user-1002.slice/user@1002.service	2	-	3.6M	-	-

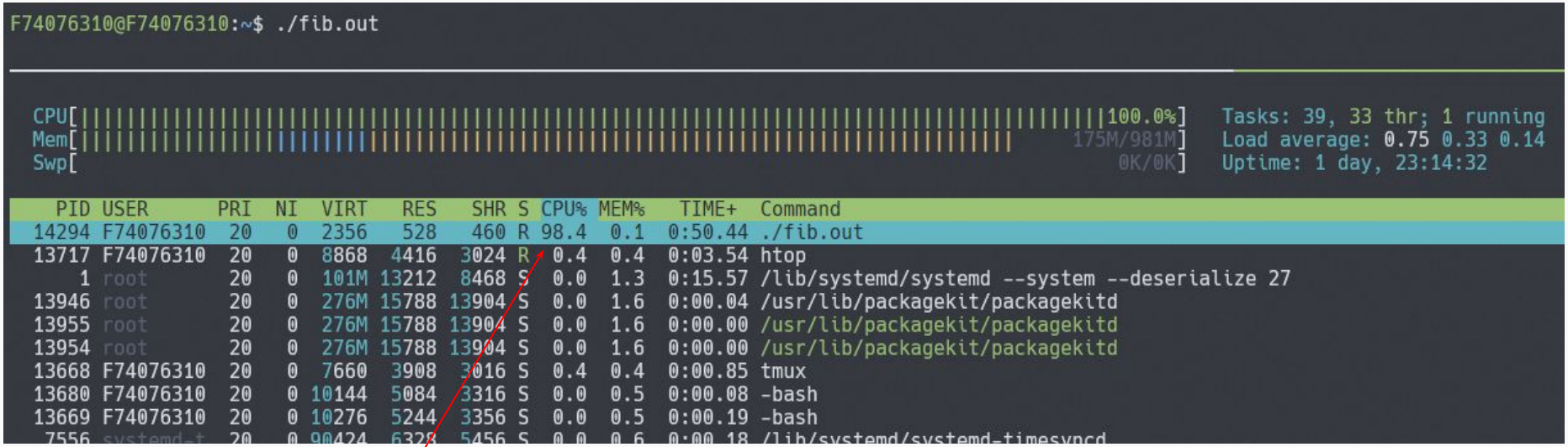
cgroups - Custom cgroups

- Define a slice
  - And add unit into slice
- Or define resource limitation in service unit file

cgroups - Demo - Fibonacci

```
F74076310@F74076310:~$ batcat fib.c
       File: fib.c
   1   #include <stdio.h>
   2   #define FIB_N 100
   3
   4   int fib(int n){
   5       if(n <= 2) return 1;
   6       else return fib(n-2) + fib(n-1);
   7   }
   8
   9   int main(void){
  10       printf("%d\n", fib(FIB_N));
  11       return 0;
  12   }
```

cgroups - Demo - Fibonacci (cont.)



Notice the high CPU usage

cgroups - Demo - Fibonacci (cont.)

```
F74076310@F74076310:~$ cat /etc/systemd/system/fib.service
[Unit]
Description=Fibonacci Program

[Service]
CPUQuota=10%
ExecStart=/usr/local/bin/fib.out

[Install]
WantedBy=multi-user.target
```

systemd.resource-control — Resource control unit settings

cgroups - Demo - Fibonacci (cont.)