Operating Systems and UNIX basics, authentication, *etc.*

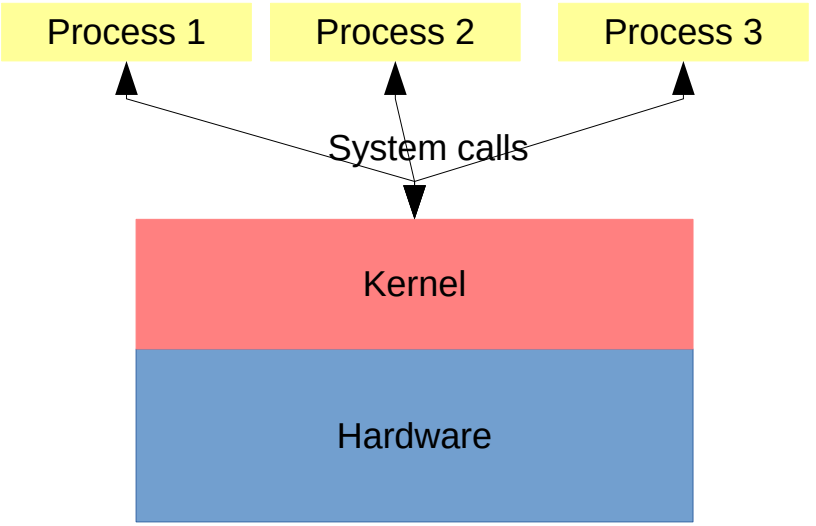
CSE 536
jedimaestro@asu.edu

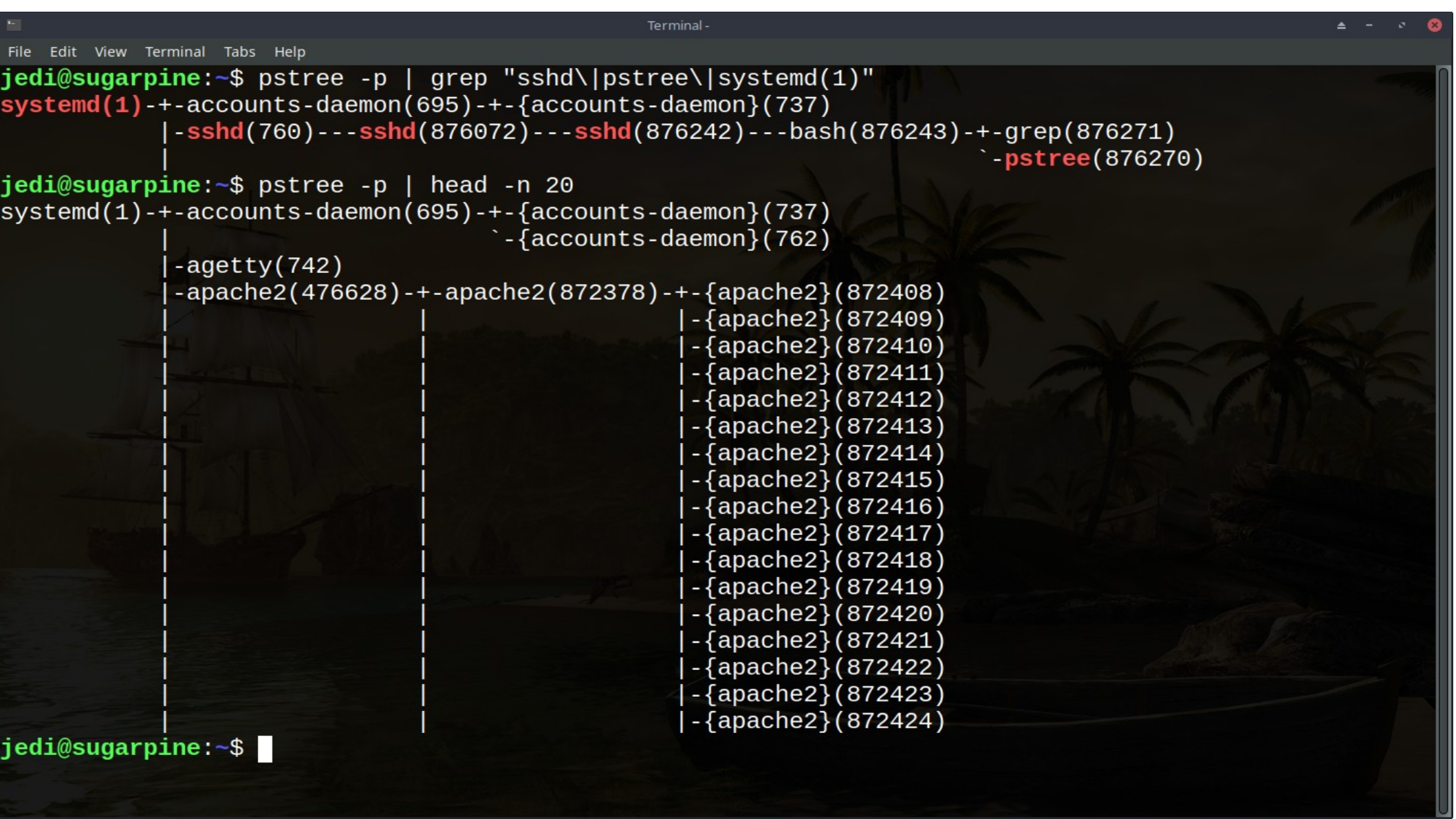
UNIX process hierarchy

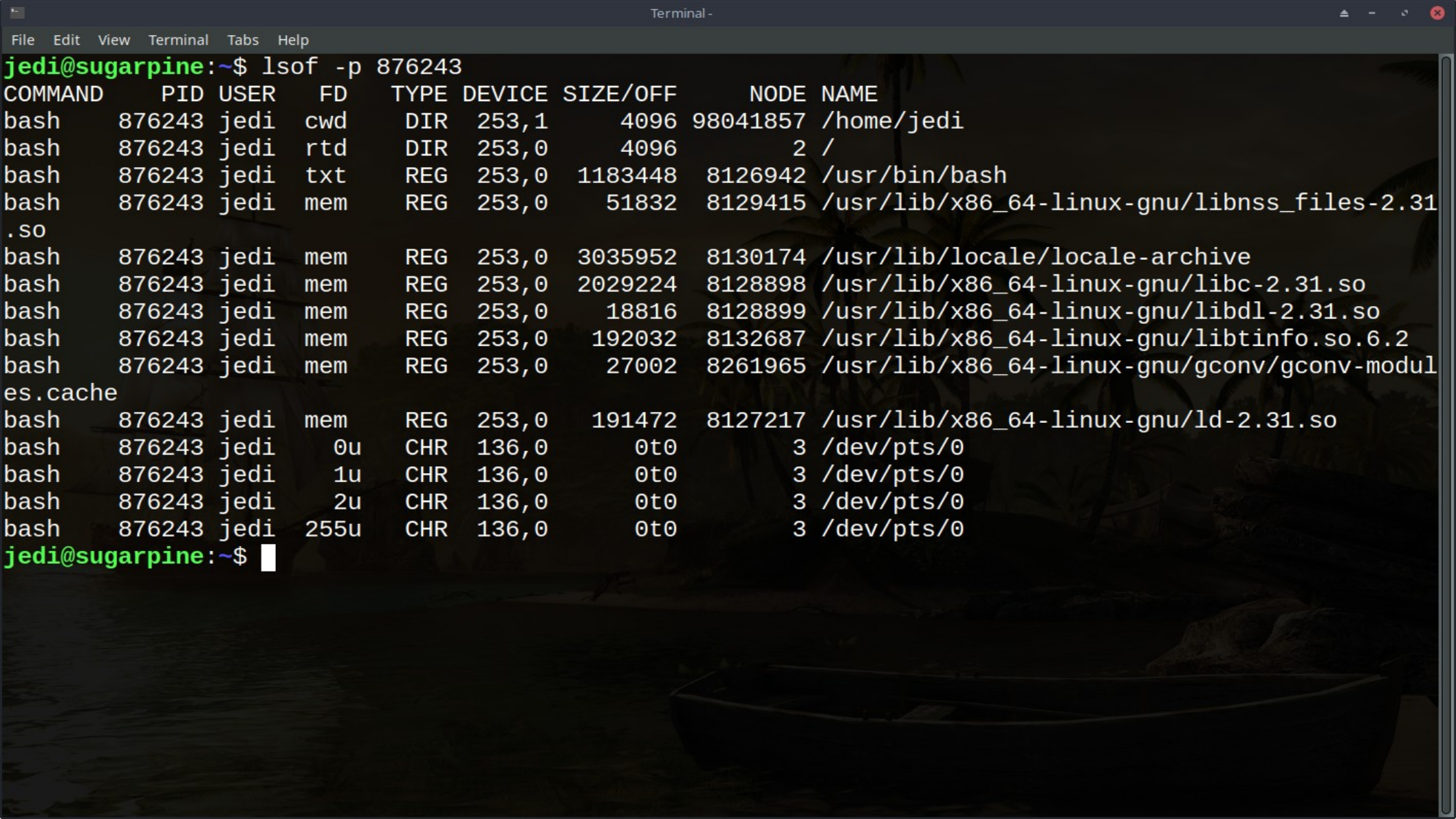
pstree -p | less -S

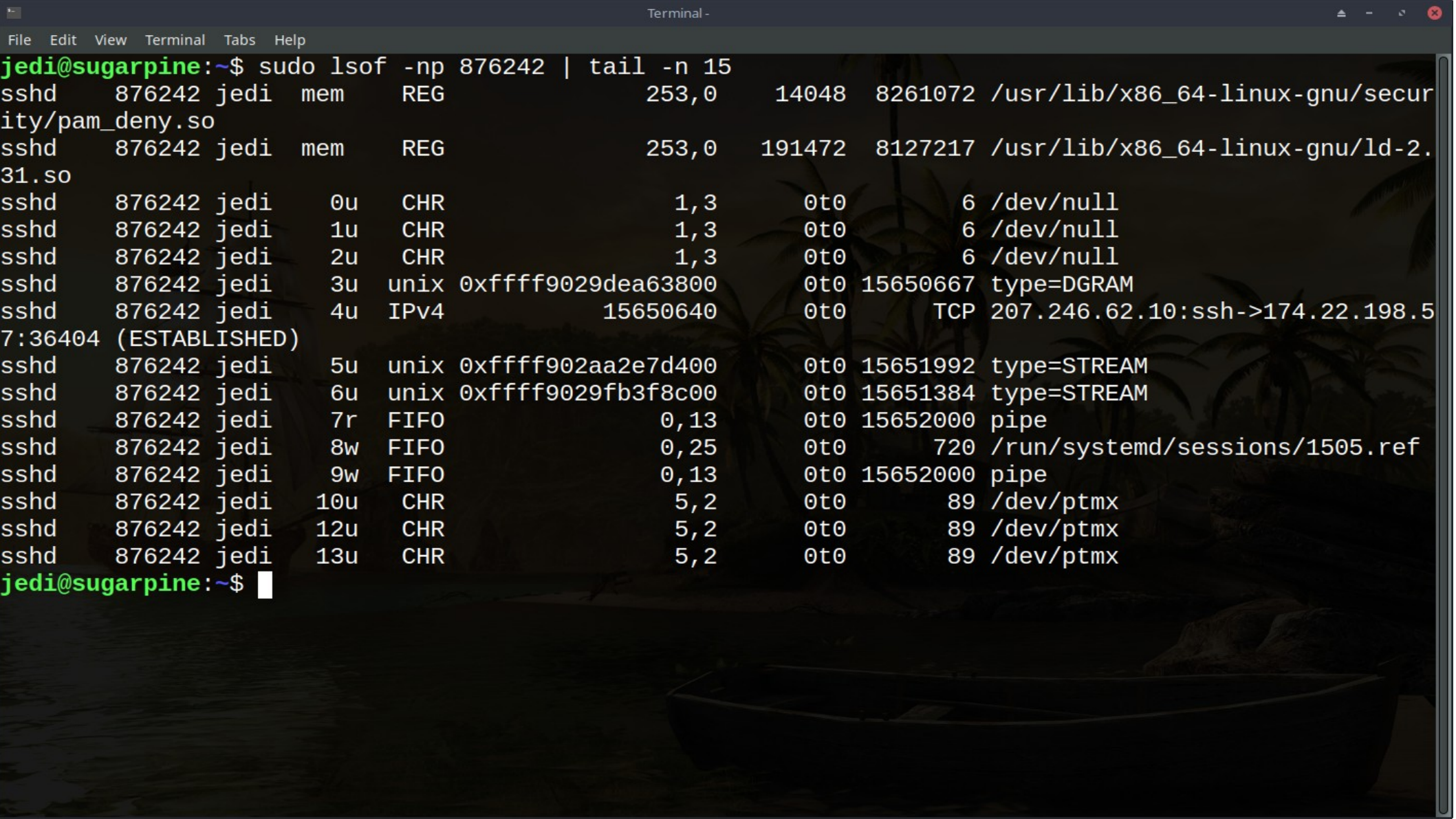
pstree -pu jedi

lsof -p 31009

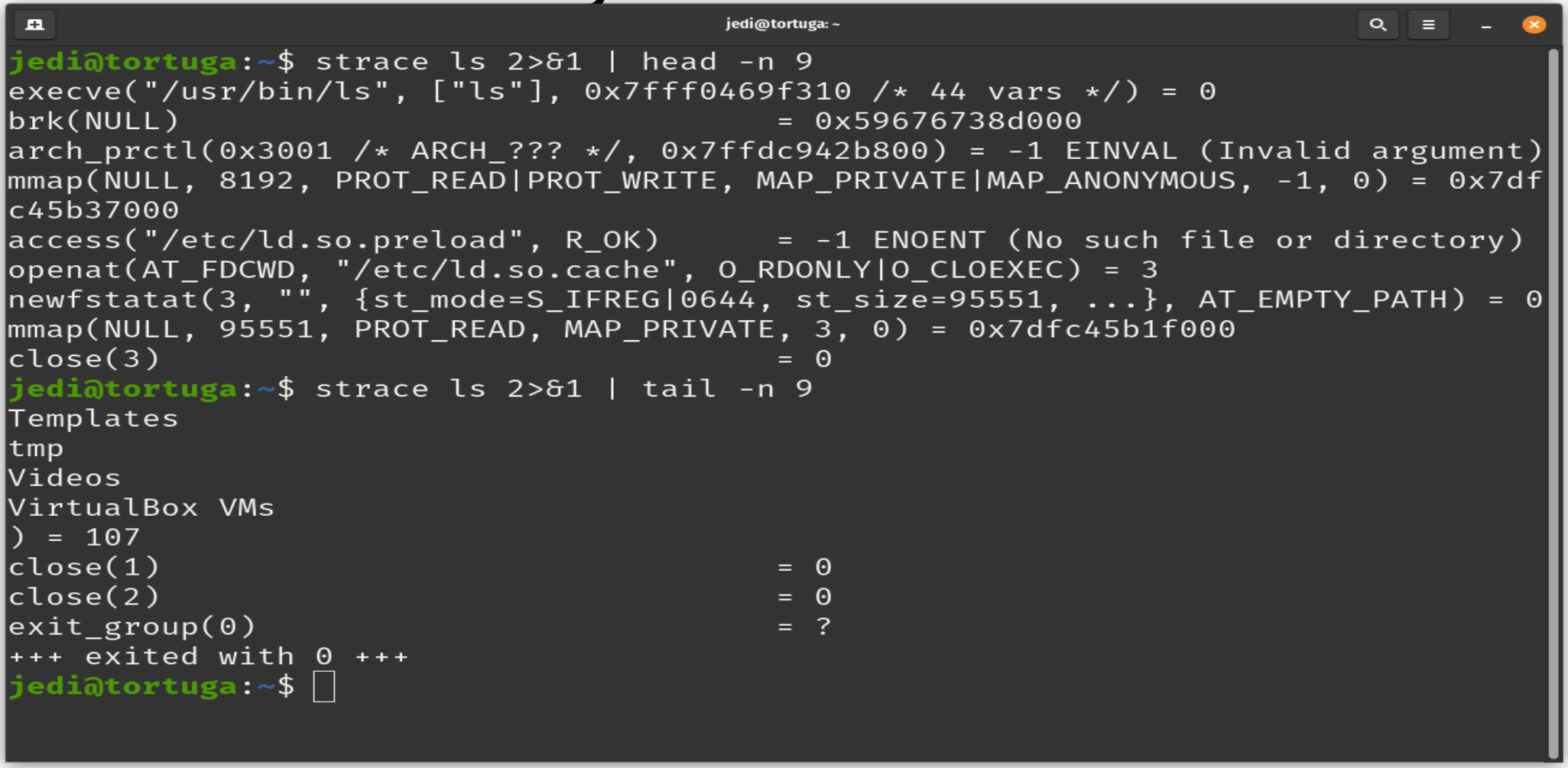



Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ pstree -p | grep "sshd\|pstree\|systemd(1)"
systemd(1)-+-accounts-daemon(695)-+-{accounts-daemon}(737)
|-sshd(760)---sshd(876072)---sshd(876242)---bash(876243)-+-grep(876271)
`-pstree(876270)
jedi@sugarpine:~$ pstree -p | head -n 20
systemd(1)-+-accounts-daemon(695)-+-{accounts-daemon}(737)
`-{accounts-daemon}(762)
|-agetty(742)
|-apache2(476628)-+-apache2(872378)-+-{apache2}(872408)
|-{apache2}(872409)
|-{apache2}(872410)
|-{apache2}(872411)
|-{apache2}(872412)
|-{apache2}(872413)
|-{apache2}(872414)
|-{apache2}(872415)
|-{apache2}(872416)
|-{apache2}(872417)
|-{apache2}(872418)
|-{apache2}(872419)
|-{apache2}(872420)
|-{apache2}(872421)
|-{apache2}(872422)
|-{apache2}(872423)
|-{apache2}(872424)
jedi@sugarpine:~$


Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ lsof -p 876243
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
bash 876243 jedi cwd DIR 253,1 4096 98041857 /home/jedi
bash 876243 jedi rtd DIR 253,0 4096 2 /
bash 876243 jedi txt REG 253,0 1183448 8126942 /usr/bin/bash
bash 876243 jedi mem REG 253,0 51832 8129415 /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
bash 876243 jedi mem REG 253,0 3035952 8130174 /usr/lib/locale/locale-archive
bash 876243 jedi mem REG 253,0 2029224 8128898 /usr/lib/x86_64-linux-gnu/libc-2.31.so
bash 876243 jedi mem REG 253,0 18816 8128899 /usr/lib/x86_64-linux-gnu/libdl-2.31.so
bash 876243 jedi mem REG 253,0 192032 8132687 /usr/lib/x86_64-linux-gnu/libtinfo.so.6.2
bash 876243 jedi mem REG 253,0 27002 8261965 /usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache
bash 876243 jedi mem REG 253,0 191472 8127217 /usr/lib/x86_64-linux-gnu/ld-2.31.so
bash 876243 jedi 0u CHR 136,0 0t0 3 /dev/pts/0
bash 876243 jedi 1u CHR 136,0 0t0 3 /dev/pts/0
bash 876243 jedi 2u CHR 136,0 0t0 3 /dev/pts/0
bash 876243 jedi 255u CHR 136,0 0t0 3 /dev/pts/0
jedi@sugarpine:~$

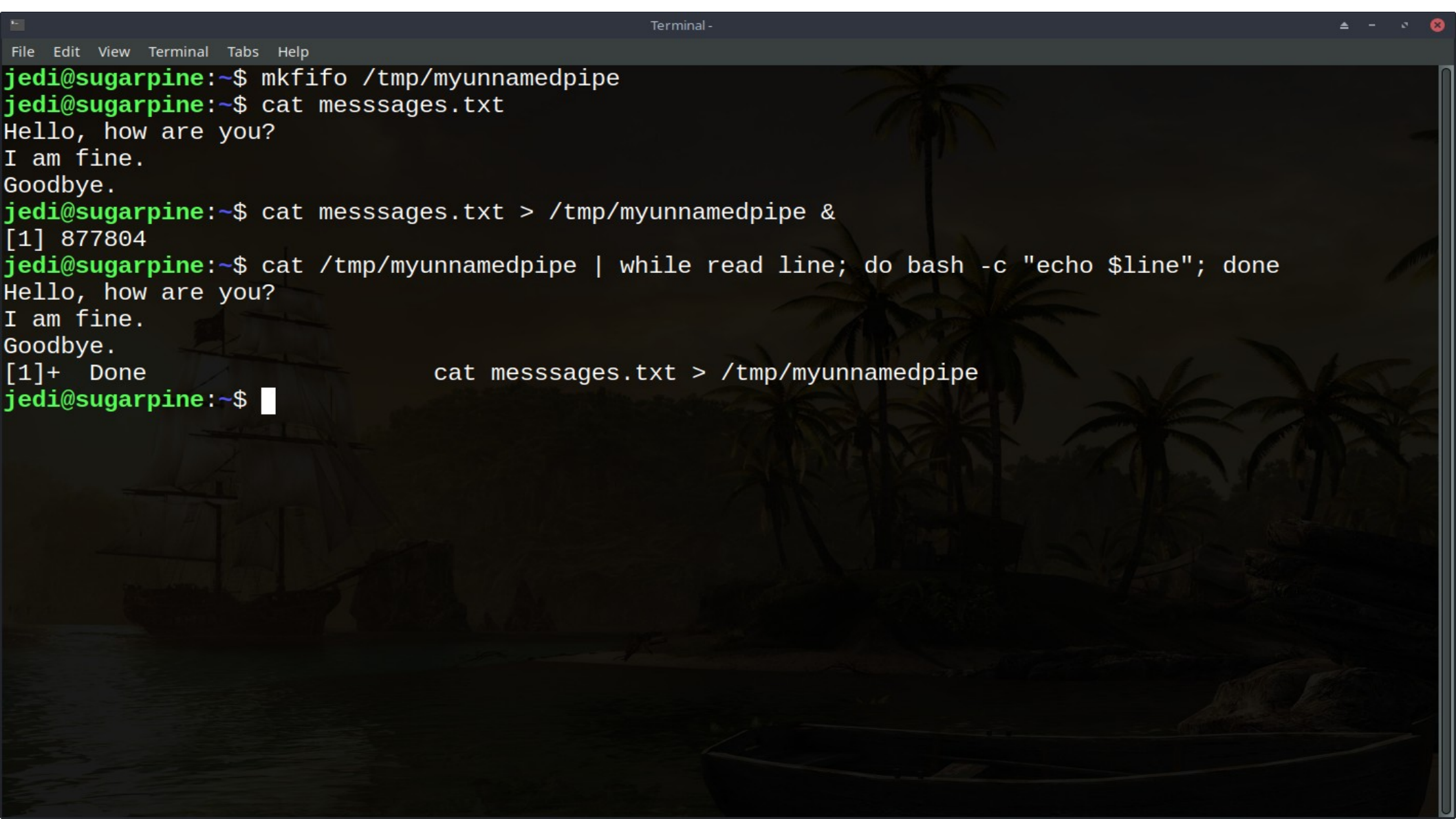

Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ sudo lsof -np 876242 | tail -n 15
sshd 876242 jedi mem REG 253,0 14048 8261072 /usr/lib/x86_64-linux-gnu/security/pam_deny.so
sshd 876242 jedi mem REG 253,0 191472 8127217 /usr/lib/x86_64-linux-gnu/ld-2.31.so
sshd 876242 jedi 0u CHR 1,3 0t0 6 /dev/null
sshd 876242 jedi 1u CHR 1,3 0t0 6 /dev/null
sshd 876242 jedi 2u CHR 1,3 0t0 6 /dev/null
sshd 876242 jedi 3u unix 0xffff9029dea63800 0t0 15650667 type=DGRAM
sshd 876242 jedi 4u IPv4 15650640 0t0 TCP 207.246.62.10:ssh->174.22.198.57:36404 (ESTABLISHED)
sshd 876242 jedi 5u unix 0xffff902aa2e7d400 0t0 15651992 type=STREAM
sshd 876242 jedi 6u unix 0xffff9029fb3f8c00 0t0 15651384 type=STREAM
sshd 876242 jedi 7r FIFO 0,13 0t0 15652000 pipe
sshd 876242 jedi 8w FIFO 0,25 0t0 720 /run/systemd/sessions/1505.ref
sshd 876242 jedi 9w FIFO 0,13 0t0 15652000 pipe
sshd 876242 jedi 10u CHR 5,2 0t0 89 /dev/ptmx
sshd 876242 jedi 12u CHR 5,2 0t0 89 /dev/ptmx
sshd 876242 jedi 13u CHR 5,2 0t0 89 /dev/ptmx
jedi@sugarpine:~$

System Calls

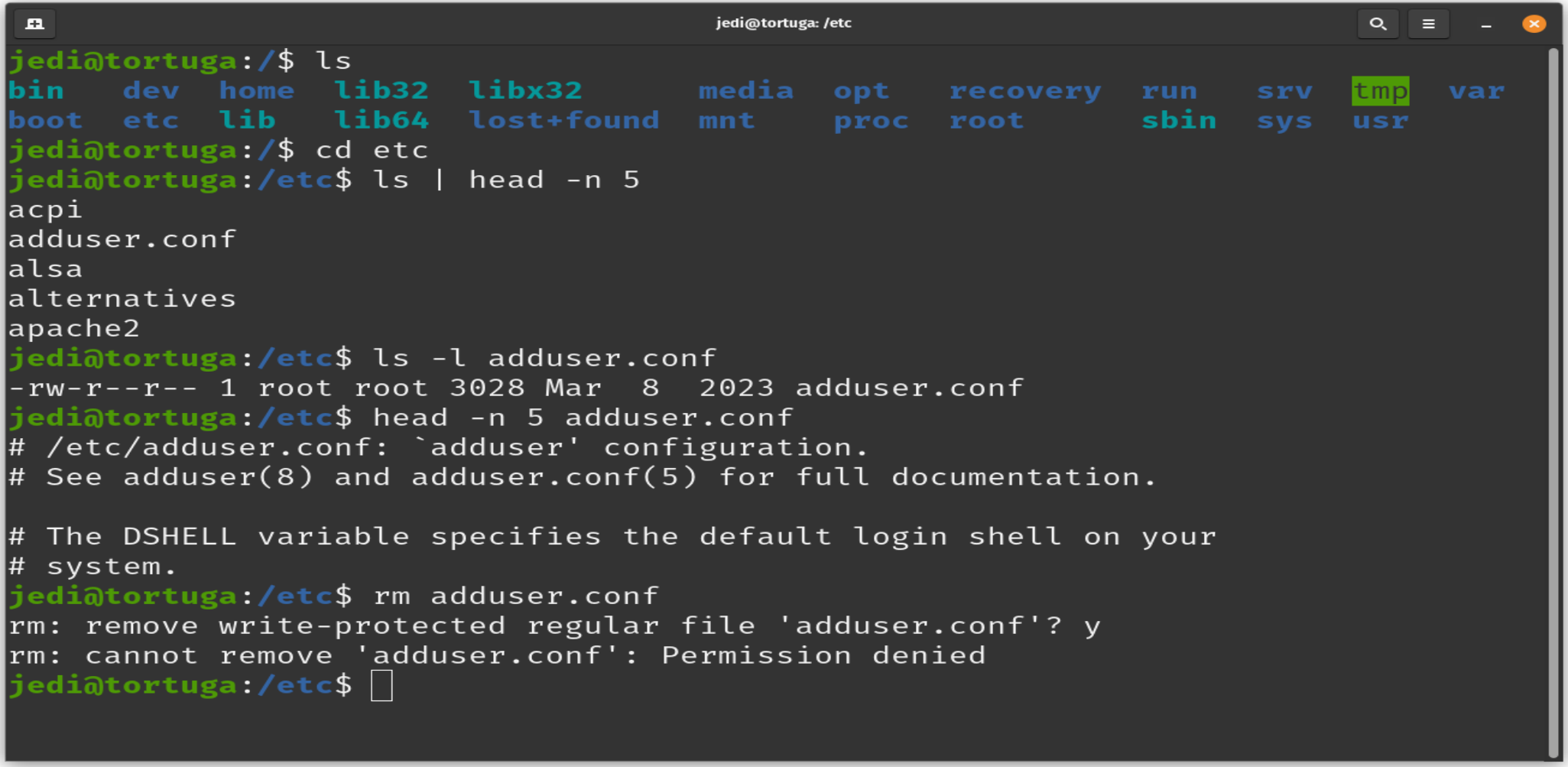


Interprocess Communication (IPC)

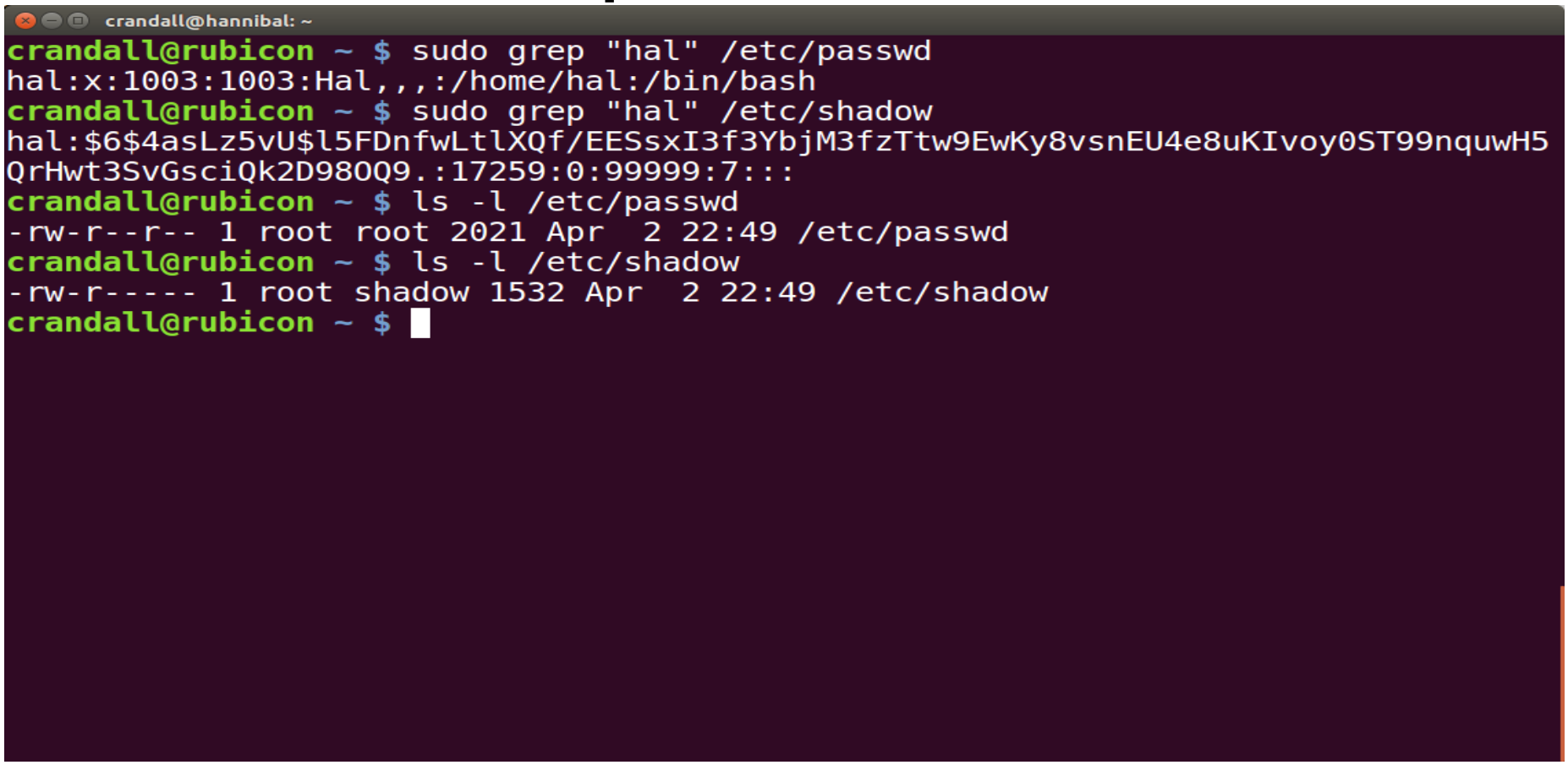
- Sockets
  - Datagram or stream
- Pipes
  - Named or unnamed
- Other ways for processes to communicate
  - Command line arguments, shared memory, file I/O, *etc.*


Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ mkfifo /tmp/myunnamedpipe
jedi@sugarpine:~$ cat messsages.txt
Hello, how are you?
I am fine.
Goodbye.
jedi@sugarpine:~$ cat messsages.txt > /tmp/myunnamedpipe &
[1] 877804
jedi@sugarpine:~$ cat /tmp/myunnamedpipe | while read line; do bash -c "echo $line"; done
Hello, how are you?
I am fine.
Goodbye.
[1]+ Done cat messsages.txt > /tmp/myunnamedpipe
jedi@sugarpine:~$

Filesystem

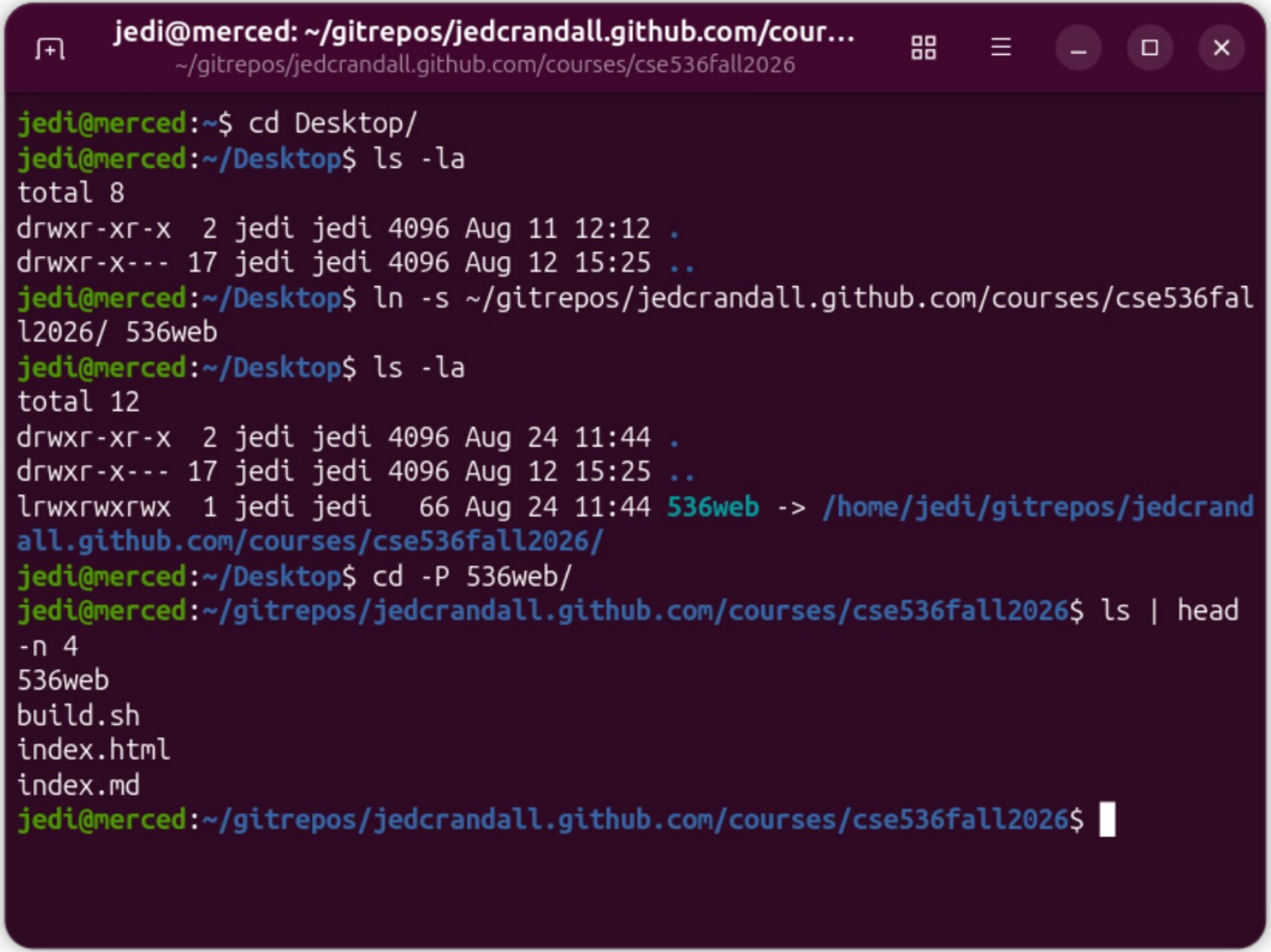

jedi@tortuga: /etc
jedi@tortuga:/$ ls
bin dev home lib32 libx32 media opt recovery run srv tmp var
boot etc lib lib64 lost+found mnt proc root sbin sys usr
jedi@tortuga:/$ cd etc
jedi@tortuga:/etc$ ls | head -n 5
acpi
adduser.conf
alsa
alternatives
apache2
jedi@tortuga:/etc$ ls -l adduser.conf
-rw-r--r-- 1 root root 3028 Mar 8 2023 adduser.conf
jedi@tortuga:/etc$ head -n 5 adduser.conf
/etc/adduser.conf: `adduser' configuration.
See adduser(8) and adduser.conf(5) for full documentation.
The DSHELL variable specifies the default login shell on your
system.
jedi@tortuga:/etc$ rm adduser.conf
rm: remove write-protected regular file 'adduser.conf'? y
rm: cannot remove 'adduser.conf': Permission denied
jedi@tortuga:/etc$

File permissions



-rwxr-x---

- First is special designations (symlink, directory)
- Next triplet is user (u)
- Triplet after is group (g)
- Last triplet is others (o)
- r = read, w = write, x = execute
- Sometimes you'll see other things, like s for Set UID


jedi@merced: ~/gitrepos/jedcrandall.github.com/cour...
~/gitrepos/jedcrandall.github.com/courses/cse536fall2026
jedi@merced:~$ cd Desktop/
jedi@merced:~/Desktop$ ls -la
total 8
drwxr-xr-x 2 jedi jedi 4096 Aug 11 12:12 .
drwxr-x--- 17 jedi jedi 4096 Aug 12 15:25 ..
jedi@merced:~/Desktop$ ln -s ~/gitrepos/jedcrandall.github.com/courses/cse536fal
l2026/ 536web
jedi@merced:~/Desktop$ ls -la
total 12
drwxr-xr-x 2 jedi jedi 4096 Aug 24 11:44 .
drwxr-x--- 17 jedi jedi 4096 Aug 12 15:25 ..
lrwxrwxrwx 1 jedi jedi   66 Aug 24 11:44 536web -> /home/jedi/gitrepos/jedcrand
all.github.com/courses/cse536fall2026/
jedi@merced:~/Desktop$ cd -P 536web/
jedi@merced:~/gitrepos/jedcrandall.github.com/courses/cse536fall2026$ ls | head
-n 4
536web
build.sh
index.html
index.md
jedi@merced:~/gitrepos/jedcrandall.github.com/courses/cse536fall2026$

Authentication in general

- Bishop, *Computer Security: Art and Science*... “Authentication is the binding of an identity to a principal. Network-based authentication mechanisms require a principal to authenticate to a single system, either local or remote. The authentication is then propagated.”

Authentication in general (continued)

- Bishop: “Authentication consists of an entity, the *user*, trying to convince a different entity, the *verifier*, of the user's identity. The user does so by claiming to know some information, to possess something, to have some particular set of physical characteristics, or to be in a specific location.”
- Informally: something you know, something you have, something you are

2FA = 2-Factor Authentication

- Two of these:
  - Something you know
  - Something you have
  - Something you are
- *E.g.*, bank card plus PIN
- For Internet services, typically the first two
- Helps protect against phishing, for example

Basic Linux authentication

- Ties you (the identity) to your user ID (the principal), which is in turn tied to subjects (*e.g.*, processes) and objects (*e.g.*, files)
- Based on hashing
  - Also salting
  - Also shadowed password hashes

username

/etc/passwd

/etc/shadow

password

Salt

SHA-512

hash

hash

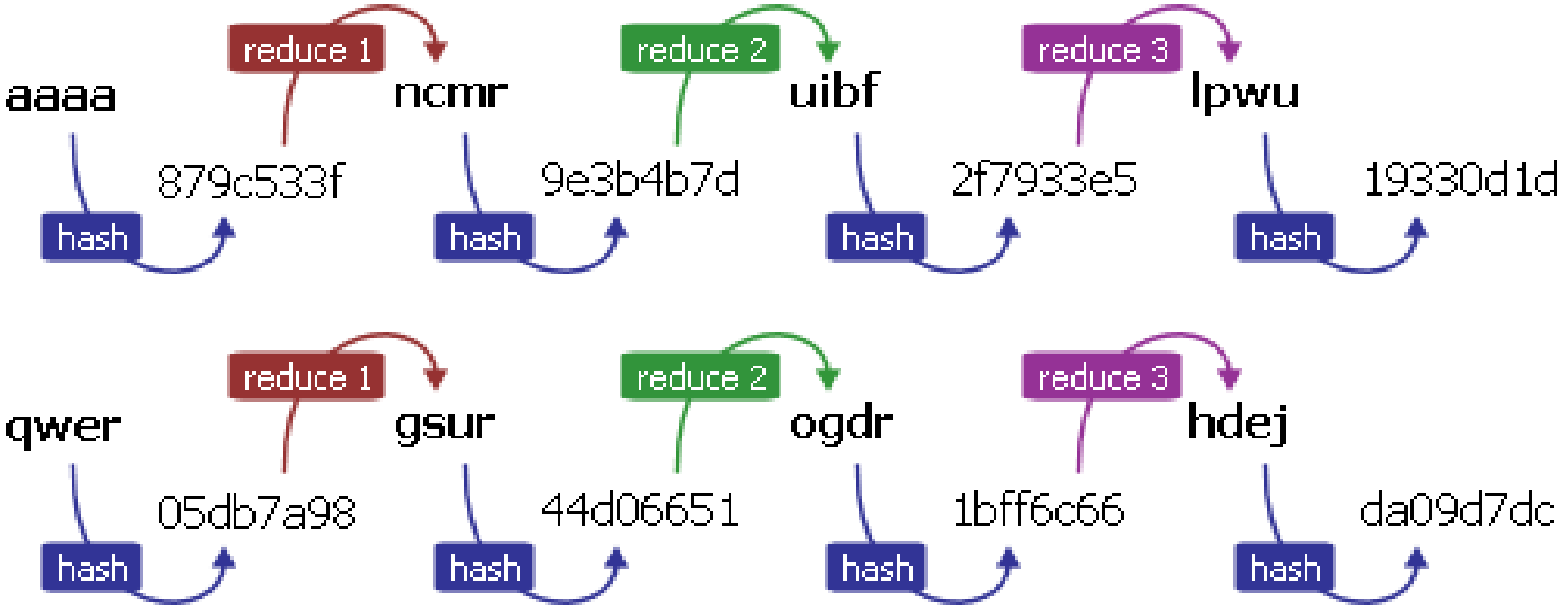
Compare

Match? Yes or no.

Passwords

- Should be high ~~entropy,~~ algorithmic complexity
- Should be easy to remember

These requirements are in
conflict with each other!
Password managers help.



Rainbow Table

aaaa	19330d1d
qwer	da09d7dc

Plagiarized from https://i.imgsafe.org/2bf87cbfe2.png

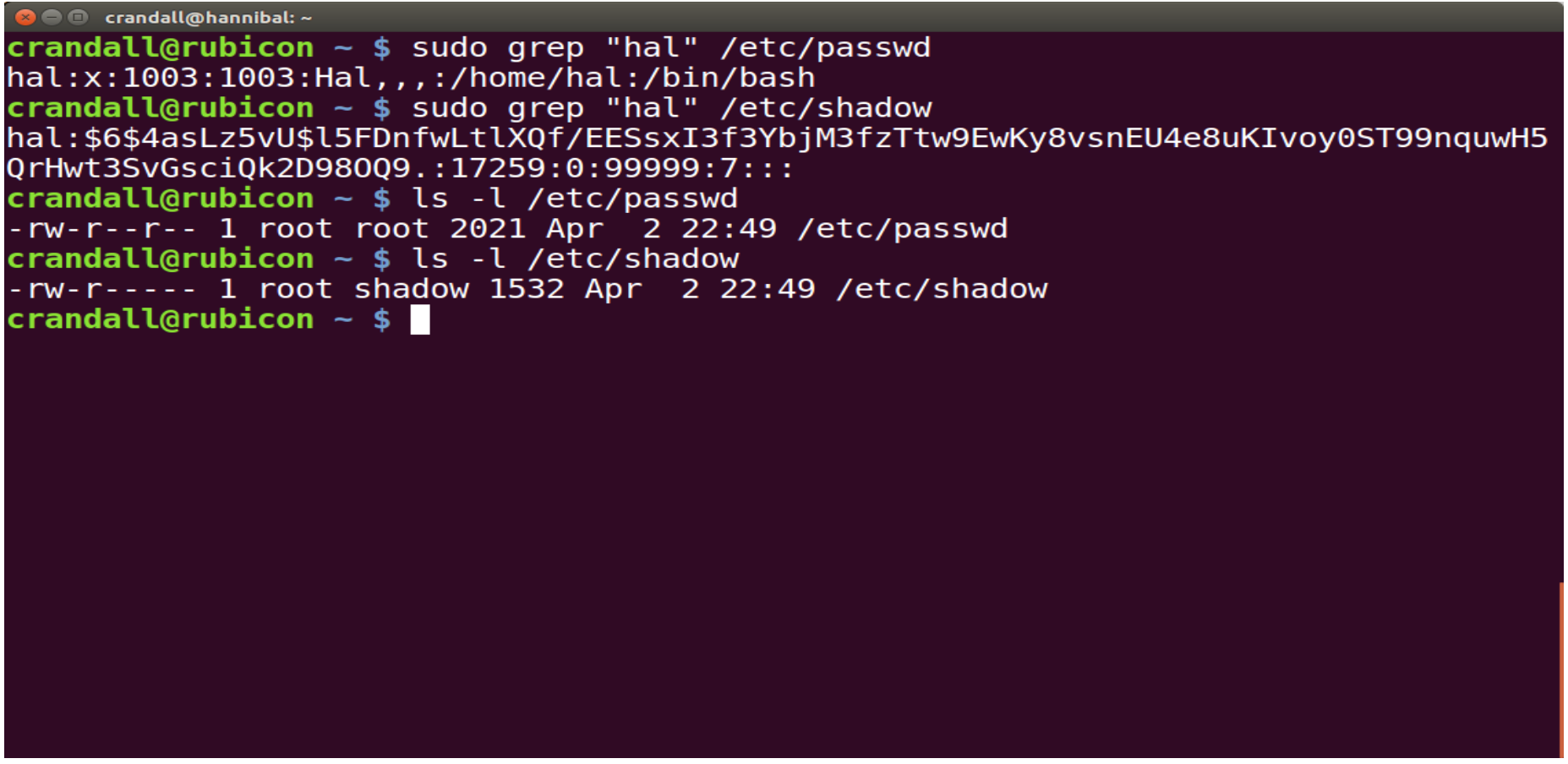
Time-memory tradeoff

- Rainbow tables can store lots of hash results compactly (precomputation)
- Just check if a user's hash might be in a hash chain, only recalculate it if so
- As a fall-back, just try every possible password (brute force)

Salting helps against precomputation.

Good passwords, system-imposed delays, shadowing help against brute force.

Shadowing the password file



What is a vulnerability?

- Management information stored in-band with regular information?
- Programming the weird machine?
- A failure to properly sanitize inputs?
- Mostly have one of two flavors:
  - One process (can be through local or IPC) sends inputs to another process that trick it into accessing or changing something it shouldn’t.
  - A process makes system calls to the kernel and tricks it in some way.

Can be local or remote, sometimes something else

- Send malicious input over a network socket to take control of a remote machine
- Give malicious input to a privileged local process to get escalated privileges for yourself
- Confuse the logic of an accounting mechanism
- Break the separation between web sites in a browser to get access to someone's bank credentials

Plagiarized from https://sites.psu.edu/thedeepweb/2015/09/17/captain-crunch-and-his-toy-whistle/

Other examples of logic bugs or more general vulnerabilities?

- Werewolves has a couple
- Amazon shopping cart (there was an IEEE Symposium on Security and Privacy paper about this, but I can't find it)
- Pouring salt water or putting tabs from construction sites in Coke machines
- Getting a code out of a locked locker
- Other examples you know of?

SQL command injection

SELECT * where username = '$u' and password = '$p'

$u = crandall
$p = abc123

SELECT * where username = 'crandall' and password = 'abc123'

SQL command injection

SELECT * where username = '$u' and password = '$p'

$u = bla' or '1' = '1' --
$p = idontknow

SELECT * where username = 'bla' or '1' = '1' --' and password = 'idontknow'

SQL command injection

SELECT * where username = '$u' and password = '$p'

$u = bla' or '1' = '1' --
$p = idontknow

SELECT * where username = 'bla' or '1' = '1' --' and password = 'idontknow'

Wassermann and Su, POPL 2006

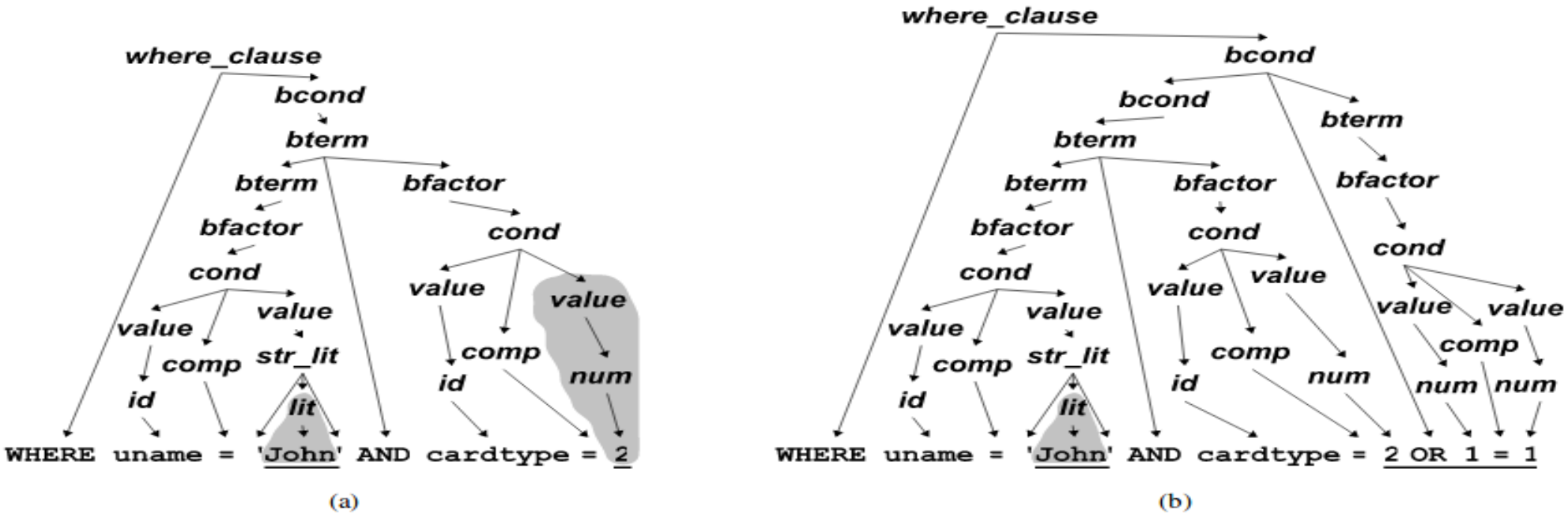
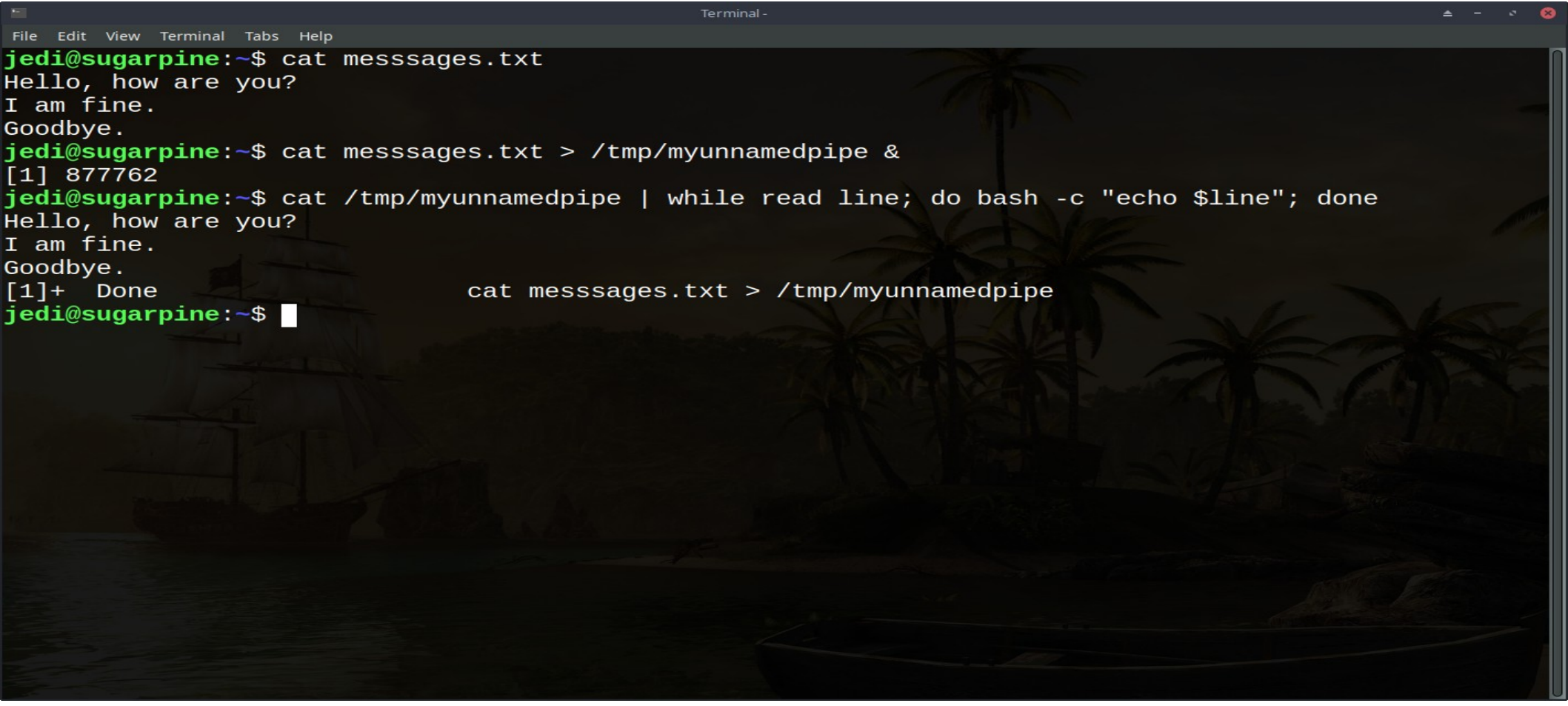


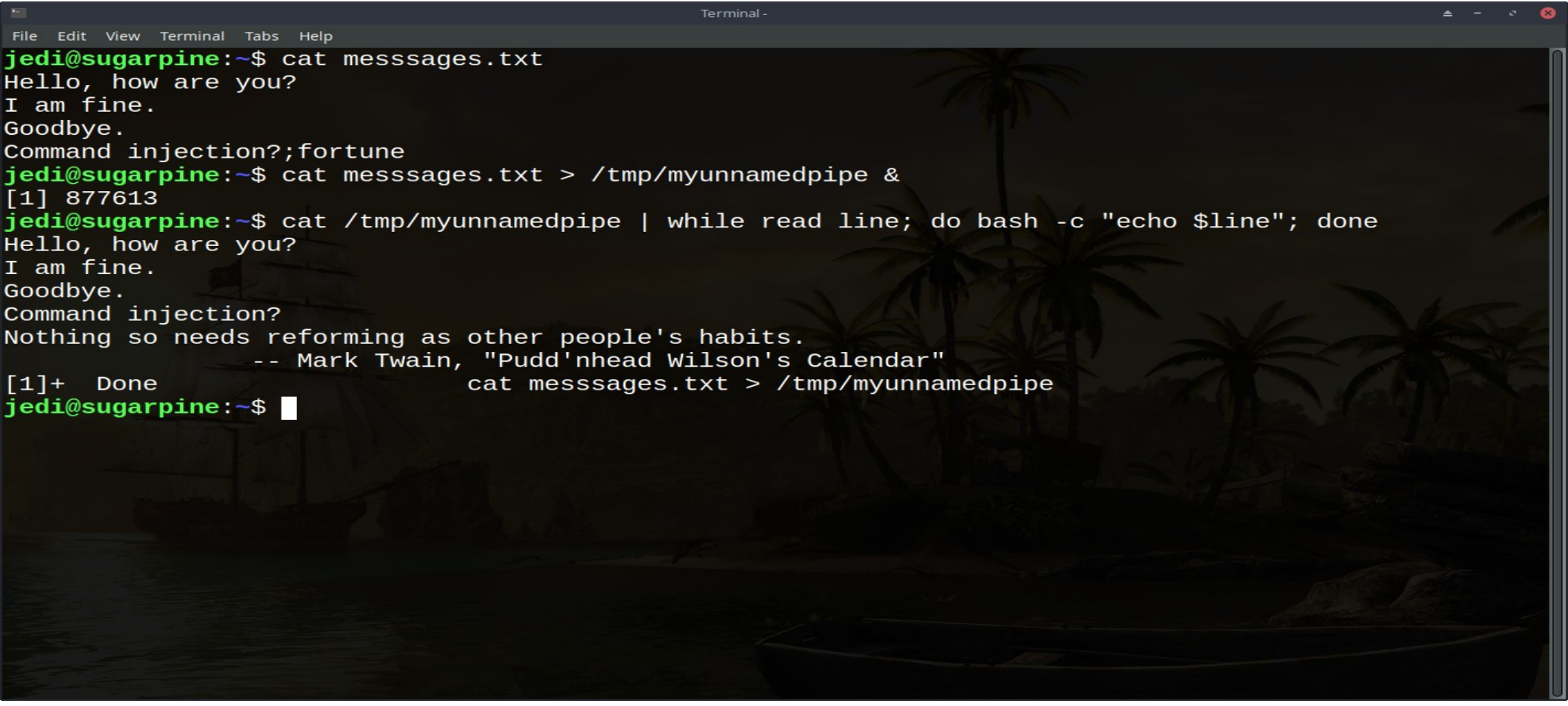
Figure 4. Parse trees for WHERE clauses of generated queries. Substrings from user input are underlined.

Cross-site Scripting (XSS)

Send a message in the WebCT platform:

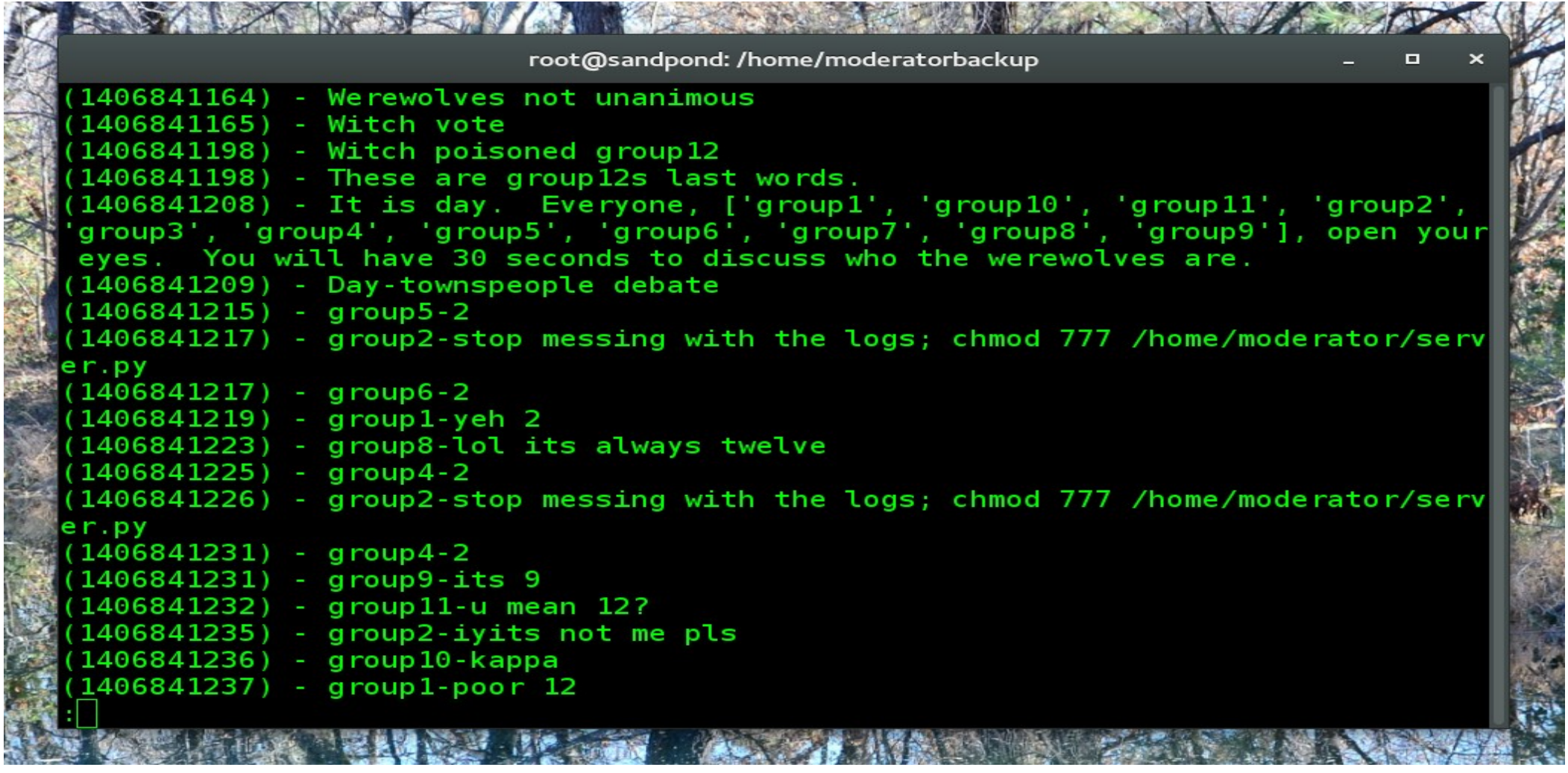
Hi Professor Crandall, I had a question about the homework. When is it due? p.s. <script>alert(“youve ben h@xored!”)</script>


Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ cat messsages.txt
Hello, how are you?
I am fine.
Goodbye.
jedi@sugarpine:~$ cat messsages.txt > /tmp/myunnamedpipe &
[1] 877762
jedi@sugarpine:~$ cat /tmp/myunnamedpipe | while read line; do bash -c "echo $line"; done
Hello, how are you?
I am fine.
Goodbye.
[1]+ Done cat messsages.txt > /tmp/myunnamedpipe
jedi@sugarpine:~$

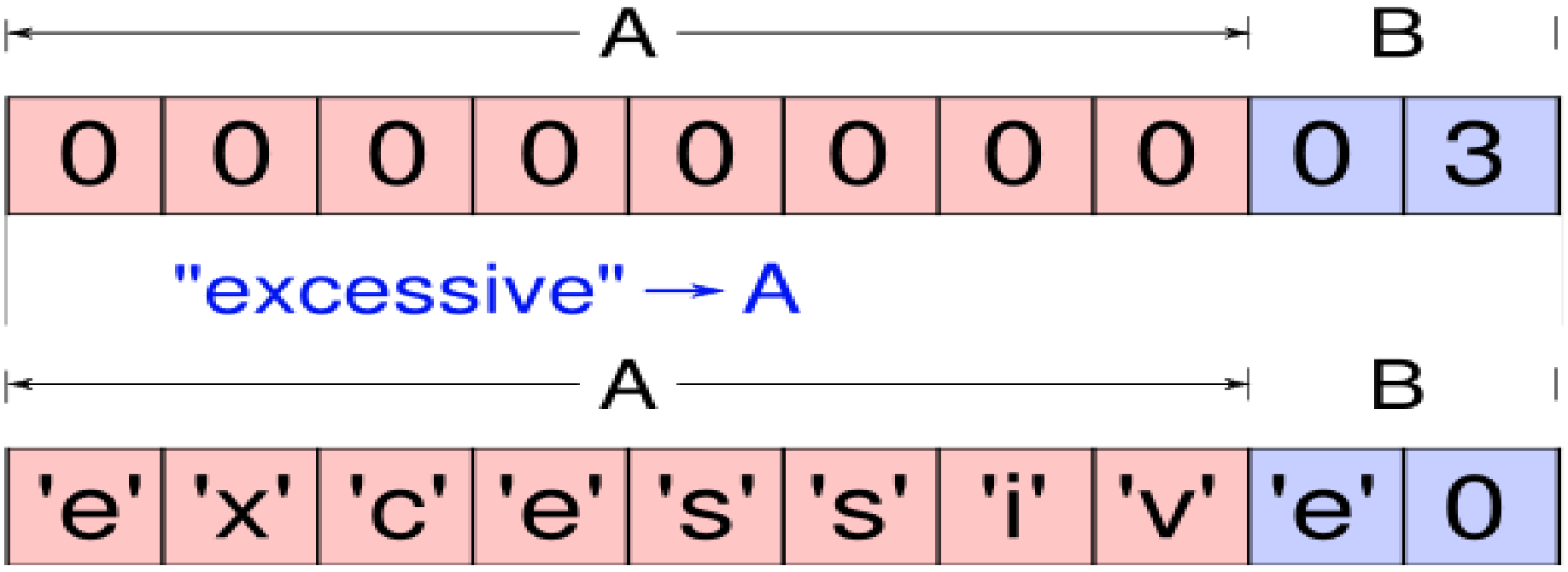

Terminal -
File Edit View Terminal Tabs Help
jedi@sugarpine:~$ cat messsages.txt
Hello, how are you?
I am fine.
Goodbye.
Command injection?;fortune
jedi@sugarpine:~$ cat messsages.txt > /tmp/myunnamedpipe &
[1] 877613
jedi@sugarpine:~$ cat /tmp/myunnamedpipe | while read line; do bash -c "echo $line"; done
Hello, how are you?
I am fine.
Goodbye.
Command injection?
Nothing so needs reforming as other people's habits.
-- Mark Twain, "Pudd'nhead Wilson's Calendar"
[1]+ Done cat messsages.txt > /tmp/myunnamedpipe
jedi@sugarpine:~$

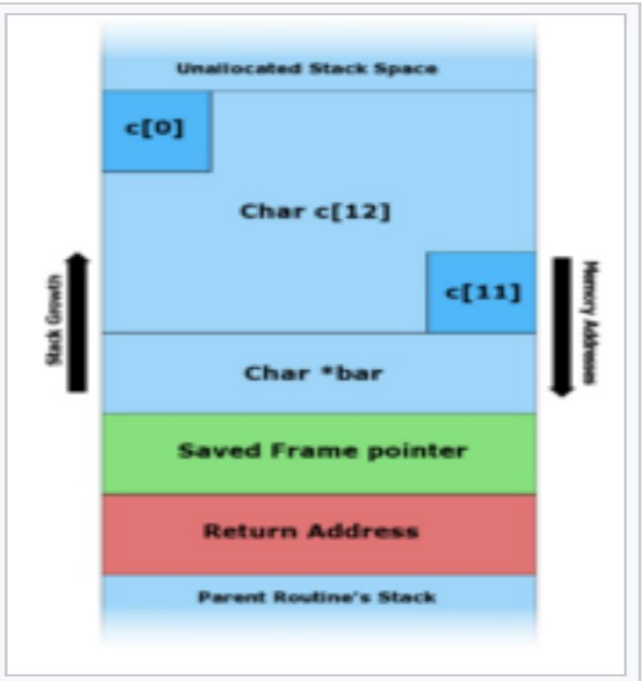
Werewolves command injection

```
system(“echo $s > /path/to/pipe“)

$s = hi; chmod 777 ~/server.py

echo hi; chmod 777 ~/server.py > /path/to/pipe
```

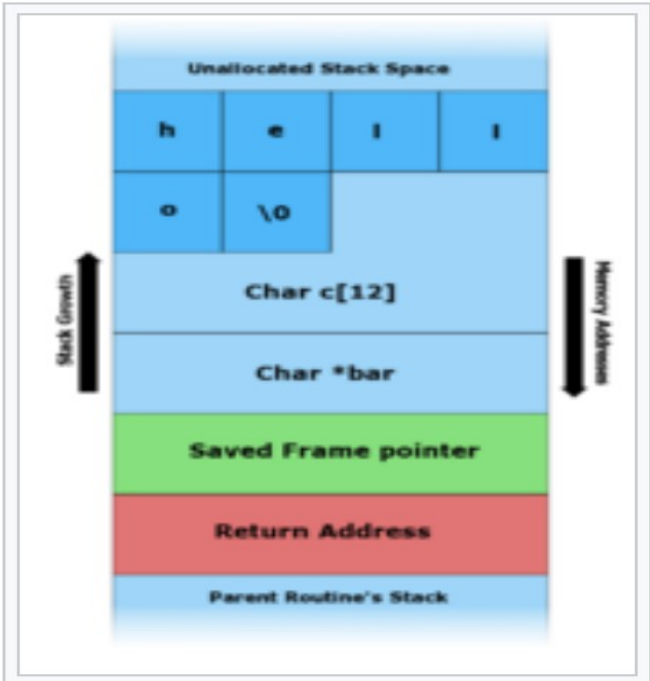

root@sandpond: /home/moderatorbackup
(1406841164) - Werewolves not unanimous
(1406841165) - Witch vote
(1406841198) - Witch poisoned group12
(1406841198) - These are group12s last words.
(1406841208) - It is day. Everyone, ['group1', 'group10', 'group11', 'group2', 'group3', 'group4', 'group5', 'group6', 'group7', 'group8', 'group9'], open your eyes. You will have 30 seconds to discuss who the werewolves are.
(1406841209) - Day-townspeople debate
(1406841215) - group5-2
(1406841217) - group2-stop messing with the logs; chmod 777 /home/moderator/server.py
(1406841217) - group6-2
(1406841219) - group1-yeh 2
(1406841223) - group8-lol its always twelve
(1406841225) - group4-2
(1406841226) - group2-stop messing with the logs; chmod 777 /home/moderator/server.py
(1406841231) - group4-2
(1406841231) - group9-its 9
(1406841232) - group11-u mean 12?
(1406841235) - group2-iyits not me pls
(1406841236) - group10-kappa
(1406841237) - group1-poor 12
:

Buffer overflows

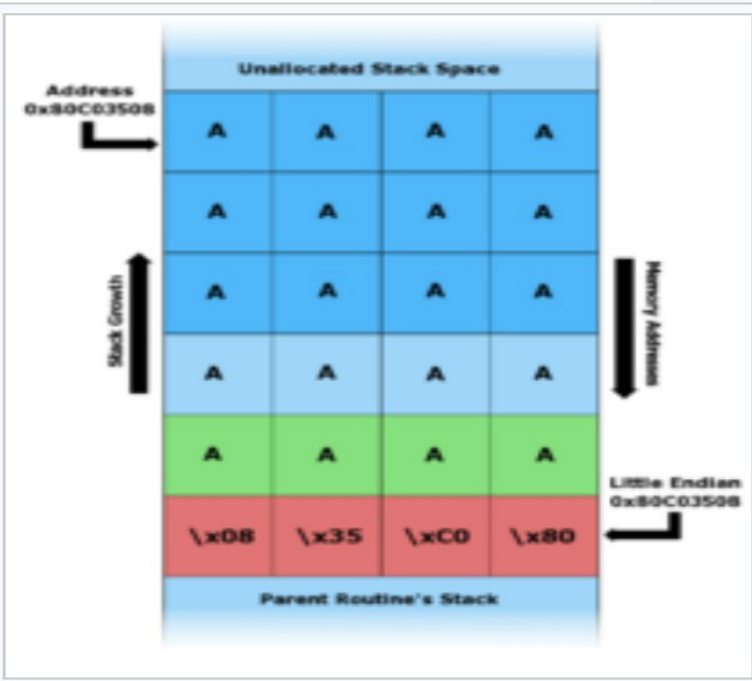




A. - Before data is copied.



B. - "hello" is the first command line argument.



C. - "AAAAAAAAAAAAAAAAAAAA\x08\x3
is the first command line argument.

https://en.wikipedia.org/wiki/Stack_buffer_overflow

Format string vulnerabilities

scanf(“%s”, string)
printf(string)

%500x%500x%12x\xbf\xff\xff\x2c%n

Memory corruption in general

- Buffer overflows on the stack and heap, format strings, double free()'s, *etc.*
- Easily the most well-studied vulnerability/exploit type
- Goal is often to execute code in memory
- See Shacham's ACM CCS 2007 paper for Return Oriented Programming
  - Even with just existing code in memory, you can build a Turing-complete machine

Race conditions

- Often called Time-of-Check-to-Time-of-Use (TOCTTOU)

```
if (!access(“/home/jedi/s”, W_OK))
{
  F = open(“/home/jedi/s”, O_WRITE);
  … /* Write to the file */
}
else
{
  perror(“You don't have permission to write to that file!”)
}
```

Werewolves race condition

touch moderatoronlylogfile.txt
chmod og-rw moderatoronlylogfile.txt

Phishing

From: "Dropbox Notification" <dropbox.noreplay@gmail.com>
Date: Dec 7, 2016 [redacted]
Subject: You have 1 new file in your inbox
To: [redacted]
Cc:

Hi [redacted]

You have received a new document in your inbox, view the file "مذكرة القبض على عزة سليمان**.pdf**" on Dropbox.

View file

Image plagiarized from https://citizenlab.org/wp-content/uploads/2017/02/Ponytail-Figure-1.png

Phishing

- Wide range of sophistication in terms of the social engineering aspect
  - One end of the spectrum: “Plez logg in and changer you password, maam!”
  - Other end of the spectrum: “The attached PDF is my notes from the meeting yesterday, it was nice to see you again!” (from someone you saw at a conference the day before)

2FA helps protect against phishing
(but state actors can easily spoof your cell phone and get SMS messages)

Coming up...

- Covert channels, where processes communicate through channels not intended for communication
  - Assumes collusion
- Side channels, where the sending process doesn’t mean to be sending
- File permissions are checked when the file is opened (and added to the file descriptor table of the process), not with every access!

man ...

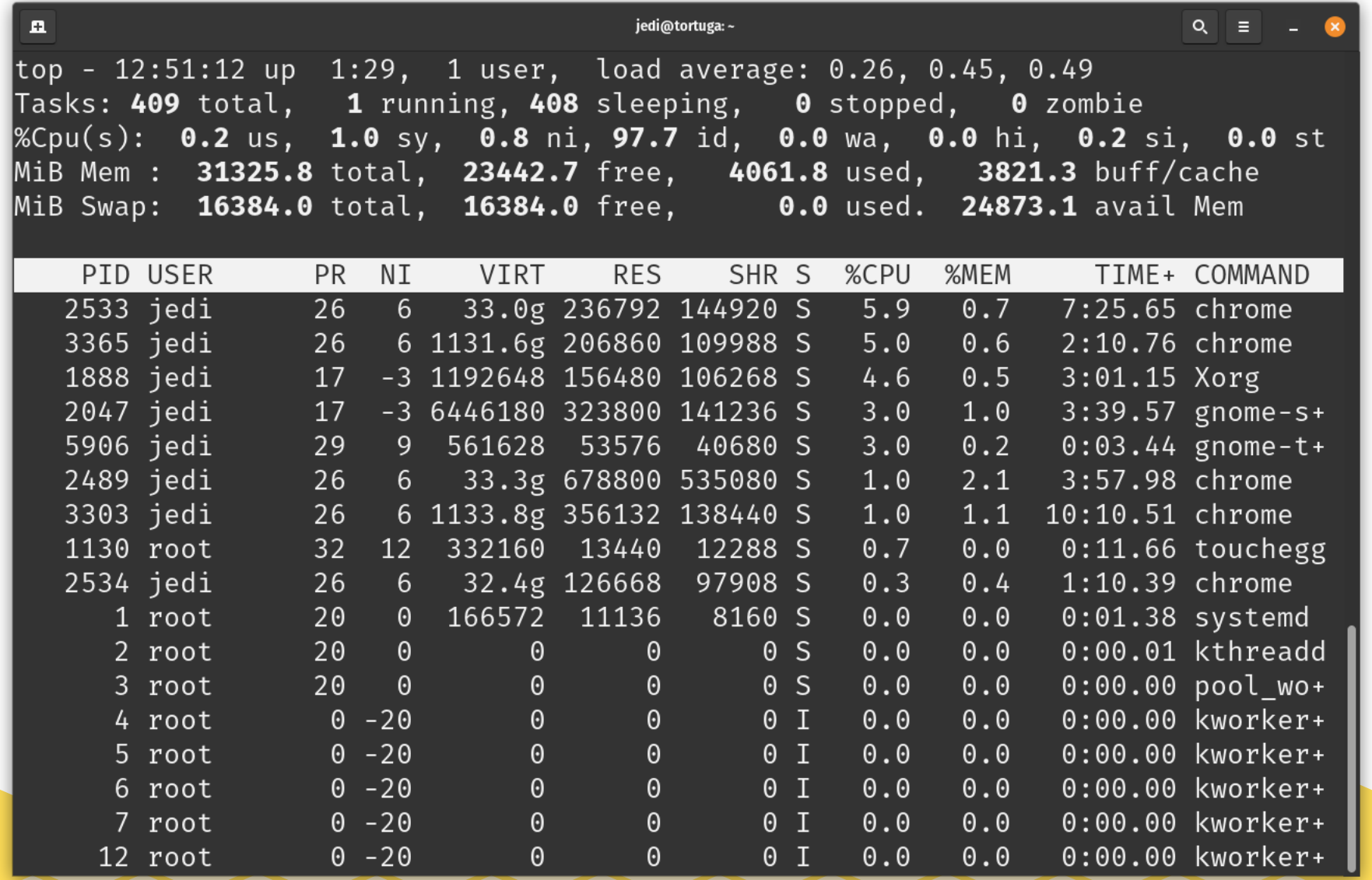
- ls (ls -l is a useful flag), cd, pwd, chown, chgrp, chmod, stat, id, w, who, last, kill, ps, pstree, netstat, cat, less, sudo, watch, screen, fuser

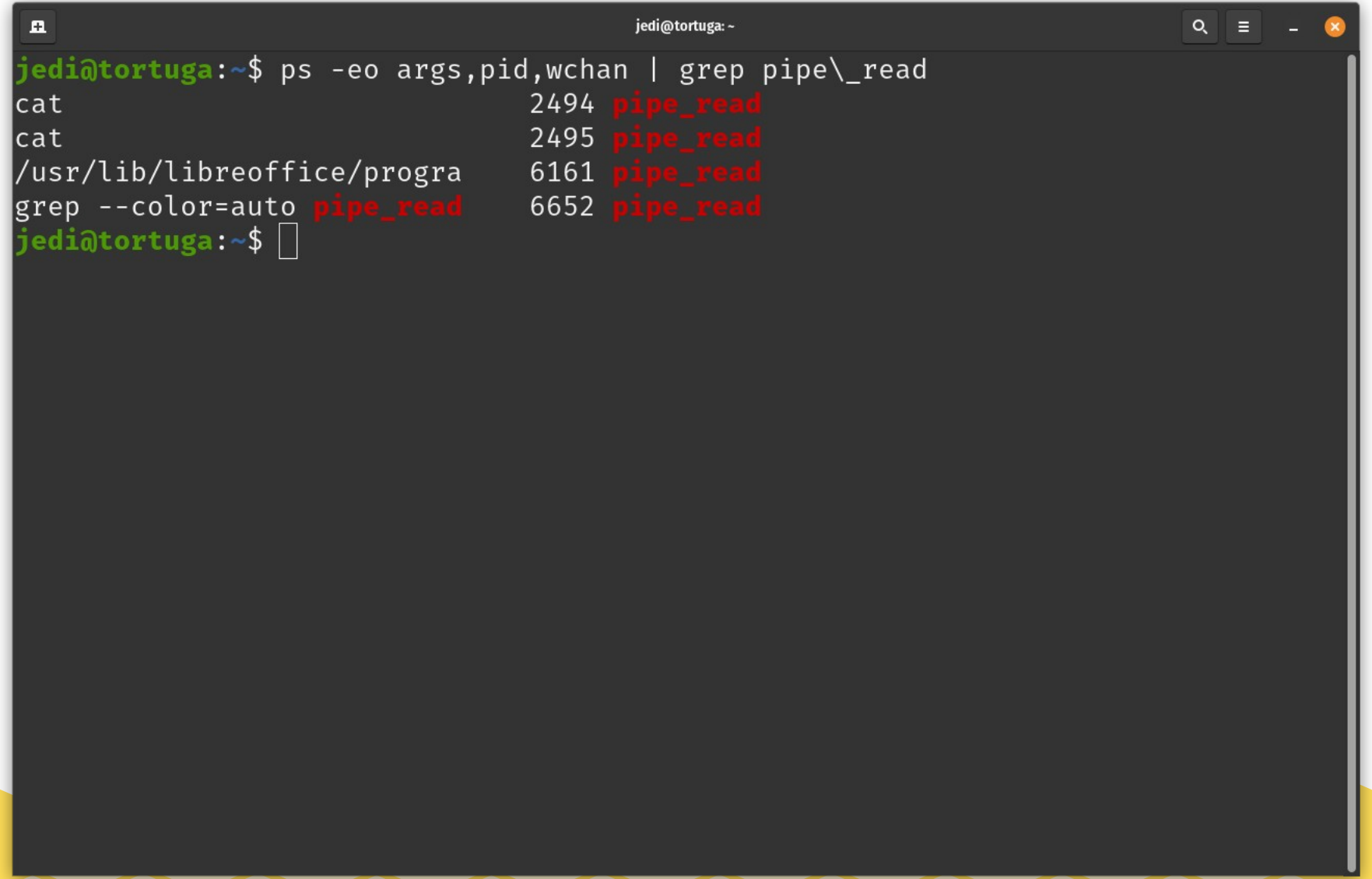
Some more things to read up on

- FIFO pipes (can be unnamed or named)
- The /proc/ filesystem
- Character devices (*e.g.*, PTY, PTS, TTY)

Resources

- *Computer Security: Art and Practice*, Chapter 12
- https://citizenlab.org/


jedi@tortuga: ~
top - 12:51:12 up 1:29, 1 user, load average: 0.26, 0.45, 0.49
Tasks: 409 total, 1 running, 408 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 1.0 sy, 0.8 ni, 97.7 id, 0.0 wa, 0.0 hi, 0.2 si, 0.0 st
MiB Mem : 31325.8 total, 23442.7 free, 4061.8 used, 3821.3 buff/cache
MiB Swap: 16384.0 total, 16384.0 free, 0.0 used. 24873.1 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2533 jedi 26 6 33.0g 236792 144920 S 5.9 0.7 7:25.65 chrome
3365 jedi 26 6 1131.6g 206860 109988 S 5.0 0.6 2:10.76 chrome
1888 jedi 17 -3 1192648 156480 106268 S 4.6 0.5 3:01.15 Xorg
2047 jedi 17 -3 6446180 323800 141236 S 3.0 1.0 3:39.57 gnome-s+
5906 jedi 29 9 561628 53576 40680 S 3.0 0.2 0:03.44 gnome-t+
2489 jedi 26 6 33.3g 678800 535080 S 1.0 2.1 3:57.98 chrome
3303 jedi 26 6 1133.8g 356132 138440 S 1.0 1.1 10:10.51 chrome
1130 root 32 12 332160 13440 12288 S 0.7 0.0 0:11.66 touchegg
2534 jedi 26 6 32.4g 126668 97908 S 0.3 0.4 1:10.39 chrome
1 root 20 0 166572 11136 8160 S 0.0 0.0 0:01.38 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kthreadd
3 root 20 0 0 0 0 S 0.0 0.0 0:00.00 pool_wo+
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
7 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+
12 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker+


jedi@tortuga: ~
jedi@tortuga:~$ ps -eo args,pid,wchan | grep pipe_read
cat 2494 pipe_read
cat 2495 pipe_read
/usr/lib/libreoffice/progra 6161 pipe_read
grep --color=auto pipe_read 6652 pipe_read
jedi@tortuga:~$

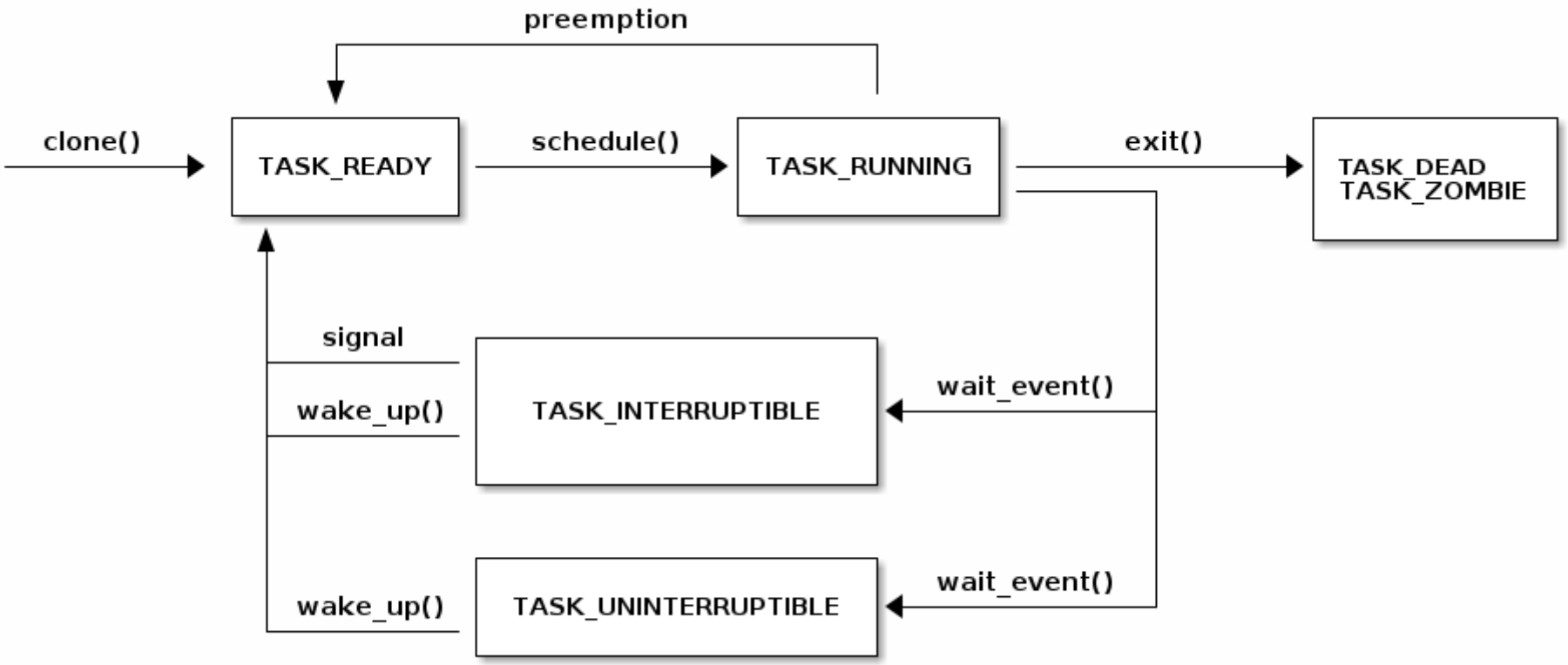
Process Control Block

- State
- Saved registers
- Address space
- File descriptor table
- Signal information
- Much more…

https://www.baeldung.com/linux/pcb

```
/* Simplified representation of the task_struct structure in Linux kernel */

struct task_struct {
    volatile long state;            // Process state (e.g., TASK_RUNNING,
TASK_STOPPED)
    struct thread_info *thread_info;
    struct exec_domain *exec_domain; // Execution domain information
(deprecated)
    struct mm_struct *mm;           // Memory management information (address
space)
    struct fs_struct *fs;           // Filesystem information
    struct files_struct *files;     // File descriptor table
    struct signal_struct *signal;   // Signal handlers and signals pending
    struct sighand_struct *sighand; // Signal handling information
    ...
    /* Various other fields */
    ...
};
```

https://linux-kernel-labs.github.io/refs/heads/master/lectures/processes.html



- TASK_INTERRUPTIBLE … Can be woken up by a signal.
- TASK_UNINTERRUPTIBLE … Can’t be woken up by a signal, *e.g.*, is waiting for some special event
  - Probably a kernel thread

fork() and exec()

```
int pid = fork()
if (pid == 0) {
    exec(“/bin/ls”);
} else {
    waitpid(pid, &status, options);
}
```

```
man fork
man clone3
```

Illusions

- Create the illusion each process has its own CPU
  - Context switch
- Create the illusion each process has its own memory
  - Virtual memory
    - Physical memory is divided into different virtual memory spaces
    - We'll discuss this more later in the semester
- Create the illusion each OS has its own physical memory and CPU
  - Virtualization

Context switches

- Reasons a CPU stops executing a process and starts executing code inside the kernel (*e.g.*, interrupt handler or scheduler)
  - Exceptions, *e.g.* …
    - Divide by zero
    - System call (could also be placed under yield)
  - Interrupts, *e.g.* …
    - I/O event
  - Yield
    - I/O request or placed on some wait queue

Simple schedulers

- FIFO, *a.k.a.*, FCFS (First In First Out, or First Come First Serve)
  - 11111111112223333
- Turnaround time
  - How long a process takes to complete
  - Assume all processes are ready in sequence 1, 2, 3 at the beginning
    - Average turnaround time = average(10, 13, 19) = 14

Simple schedulers (continued...)

- Shortest Job First
  - 22233333311111111111
  - Turnaround time improved
    - Average(3, 9, 19) = 10.333... (less than 14)

So, why not use Shortest Job First?

Reason #1

- We can’t see into the future

Reason #2

- Without *preemption*, it's hard to get a good response time
  - Response time: How long it takes the CPU to respond to a request made by a process
    - *E.g.*, you press a key, you'd like to see that letter on the screen

Add a timer interrupt...

- …that, *e.g.*, goes off every 10ms or 1ms
- Gives the scheduler a chance to schedule a new process, *e.g.*, if there's been some input and they're out of the wait queue
- Round Robin scheduler
  - Divide CPU time into slices
  - E.g., each process gets two time slices
    - 1122331123311331111

Can do even better...

- If we could see into the future?
  - Could improve response time by prioritizing (*i.e.*, letting them skip in line) processes that are very likely to yield the CPU quickly
- Can predict future behavior based on past behavior

Multilevel Feedback Queue

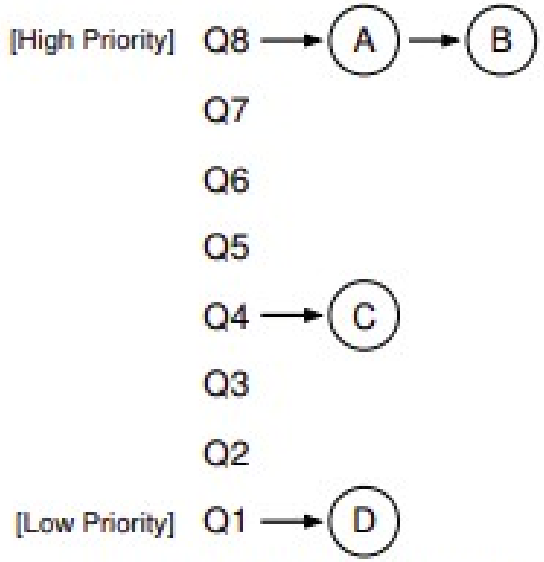


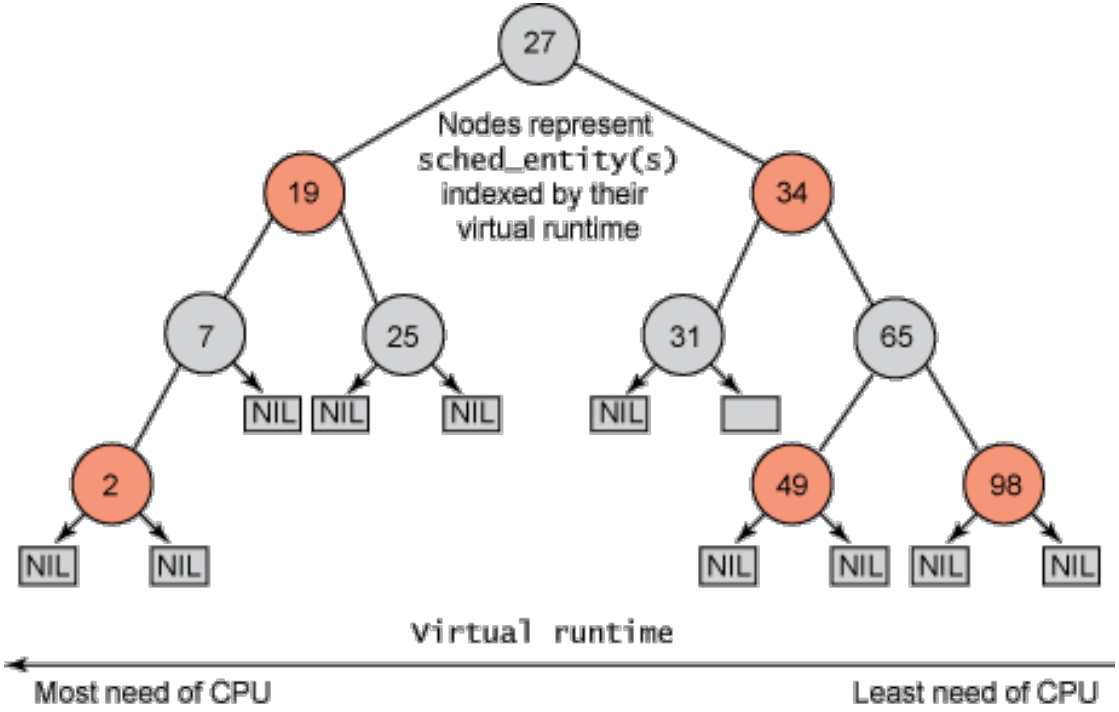
Figure 8.1: **MLFQ Example**

https://pages.cs.wisc.edu/~remzi/OSTEP/cpu-sched-mlfq.pdf

MLFQ

- Fernando Corbato's Turing award is based on this
- Famously used in Solaris
- Modern schedulers are not always MLFQ's but are based on the same ideas
- Priorities can also have a static element to them, in general
  - `man nice`
- What about starvation?

Linux Completely Fair Scheduler

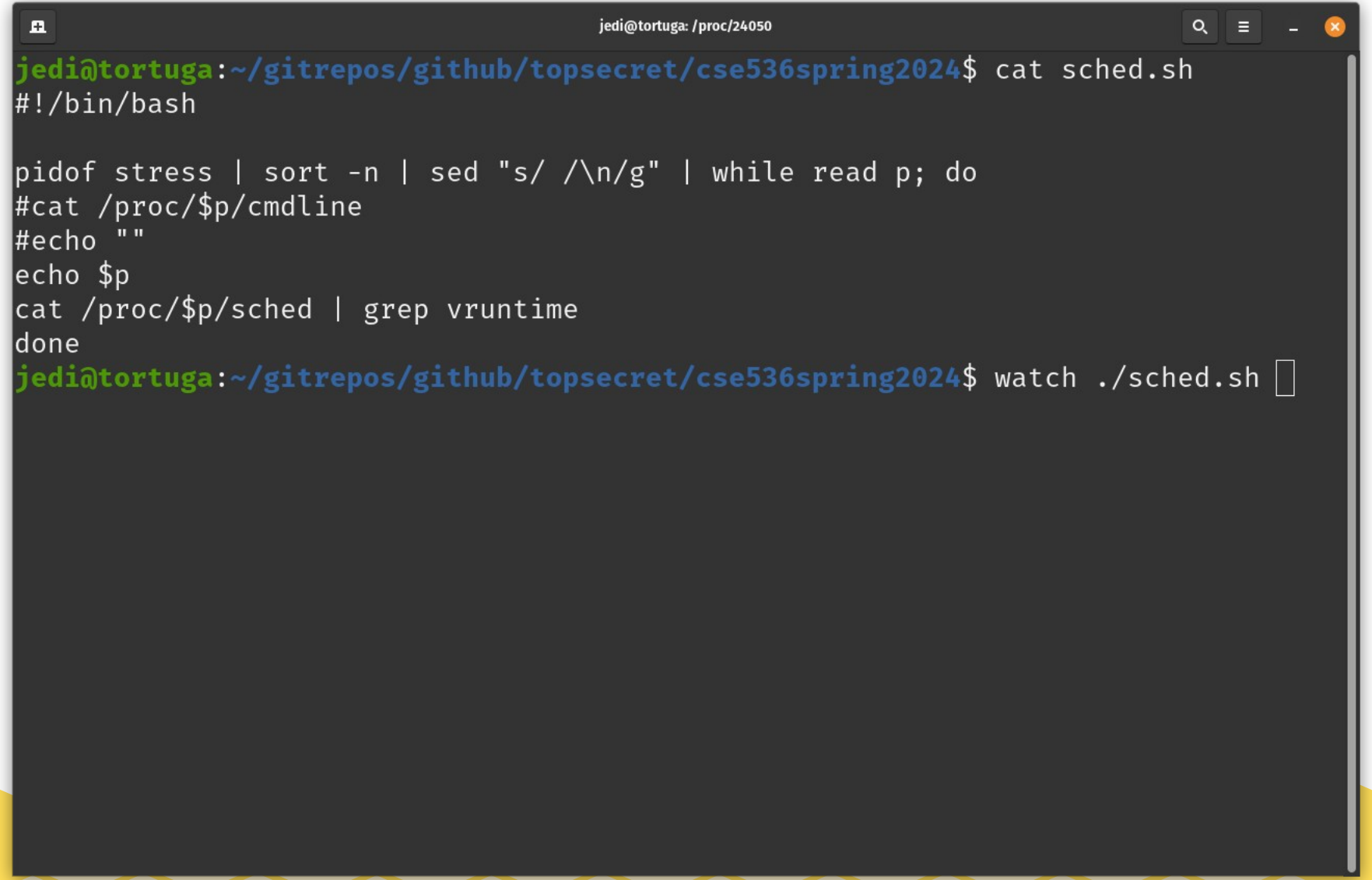


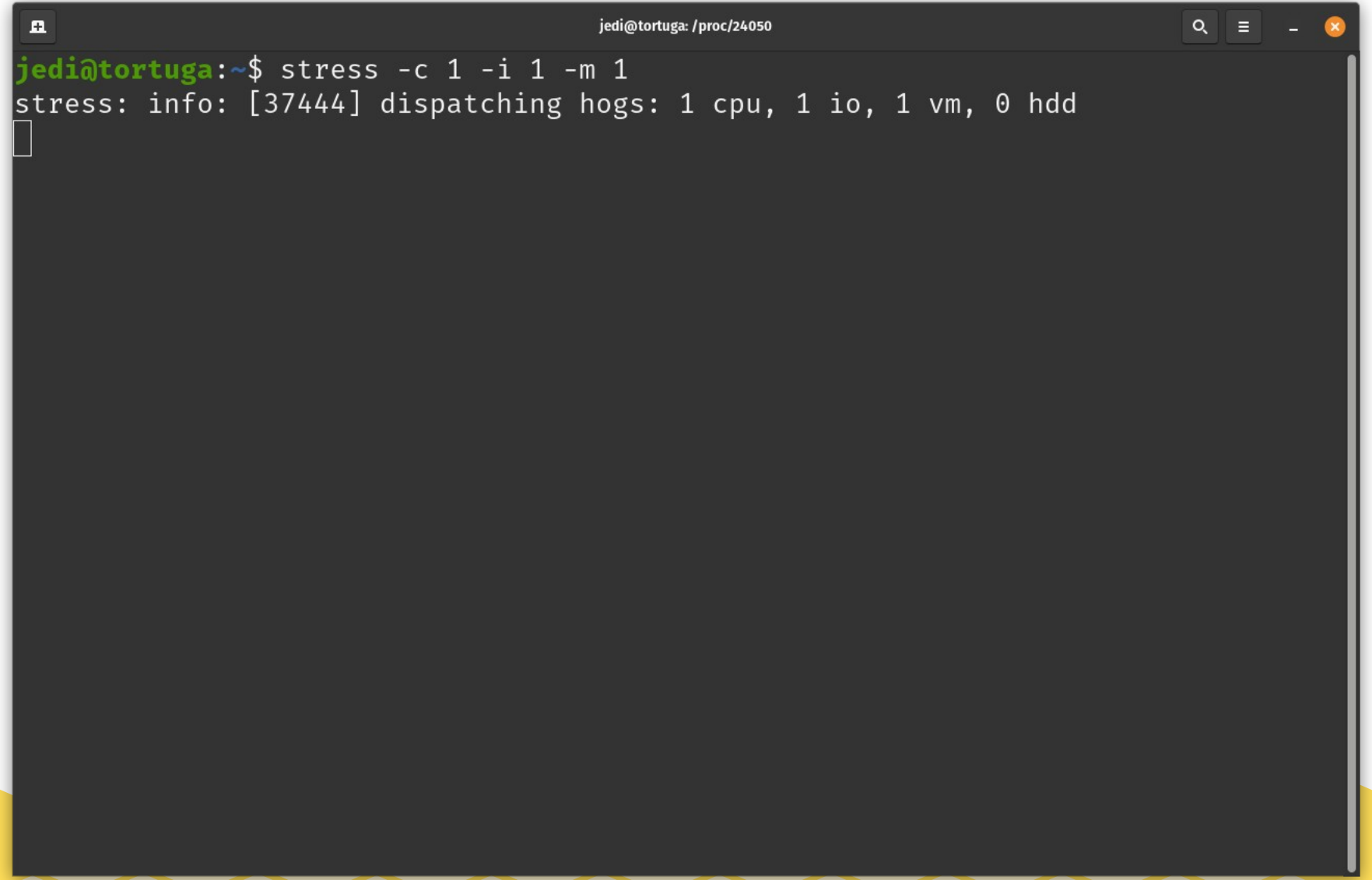
https://svalaks.medium.com/linux-internals-completely-fair-scheduling-cfs-cpu-scheduler-algorithm-7412c08d2e37

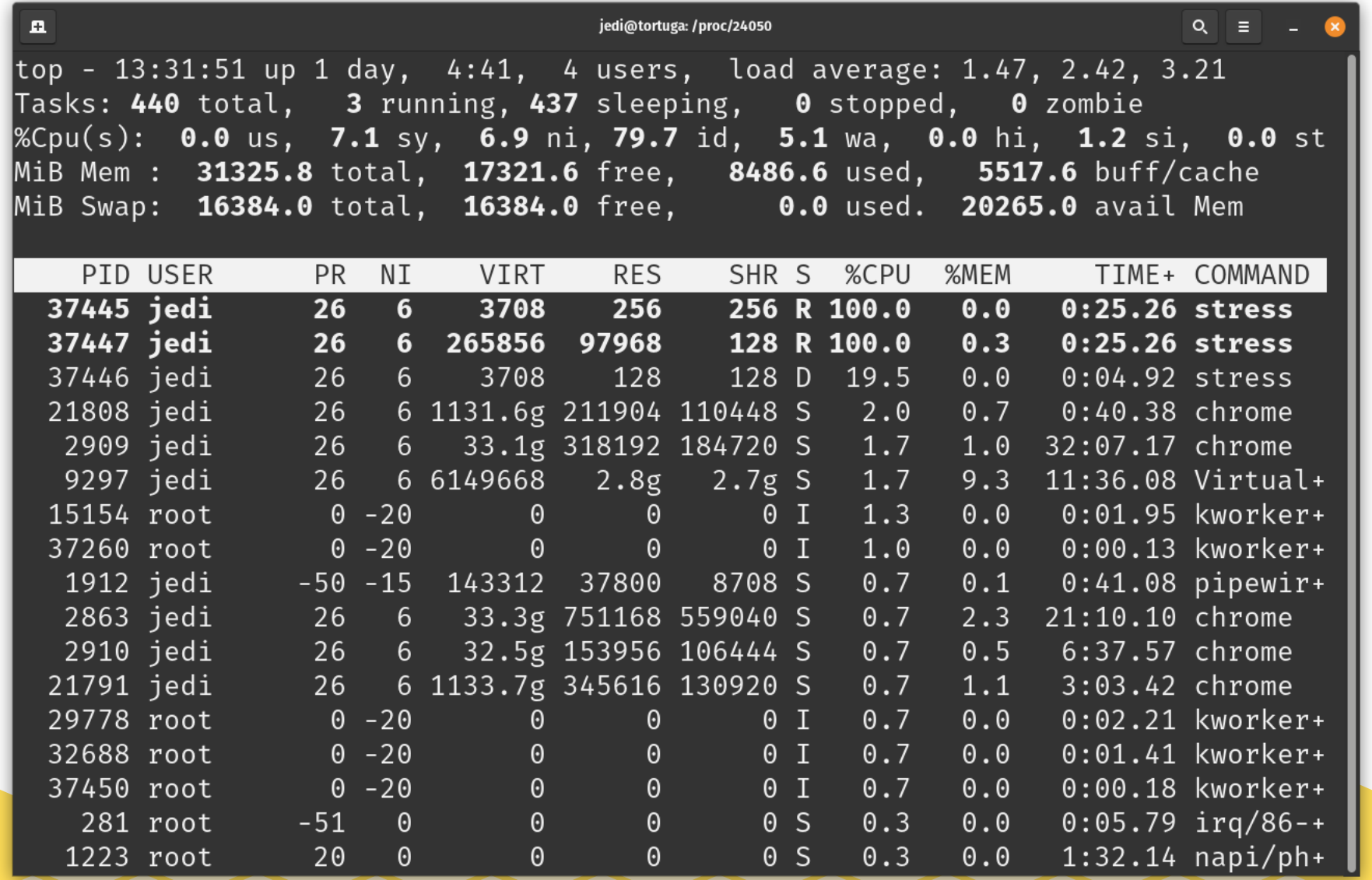
Linux CFS

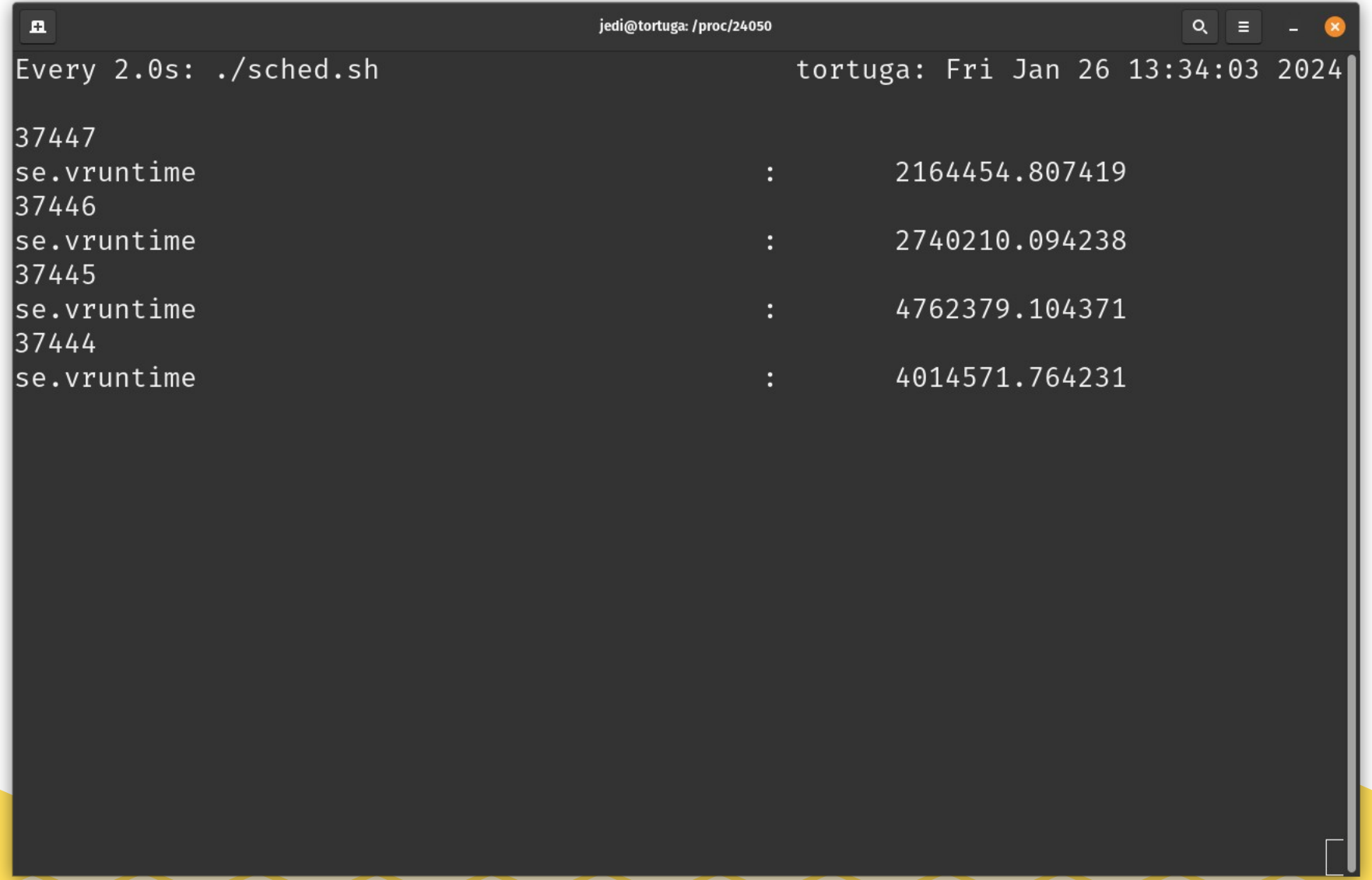
- Picks process from left-most node in O(1) time
- Reinserts when a process is done in O(log(N)) time
- The more you yield the CPU, the more you stay to the left
- The more CPU you hog, the more you move to the right
- Priority is also part of the slice calculation
- Good tradeoff of throughput and responsiveness, no starvation

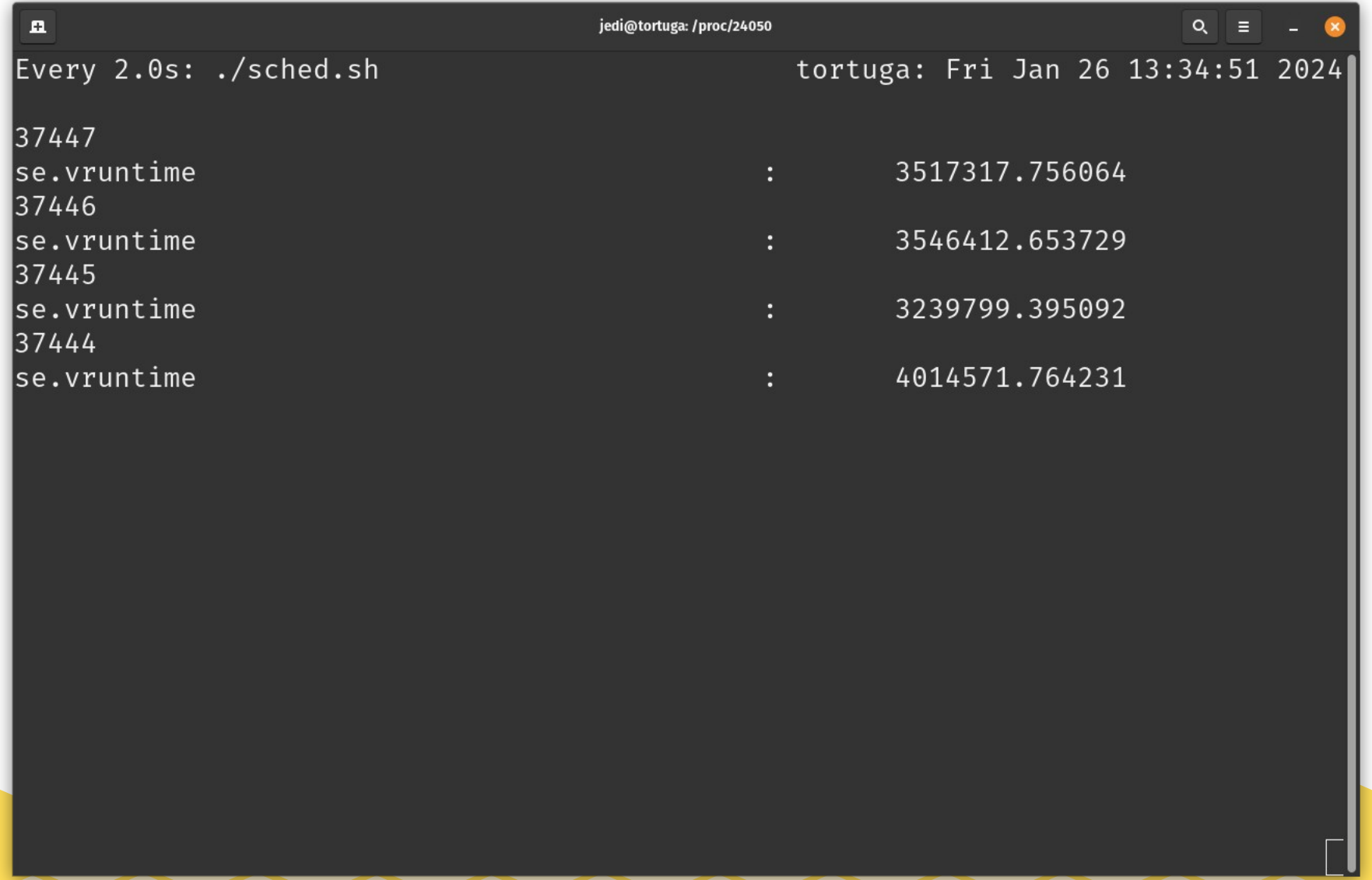
CFS is now outdated, we'll cover Linux's EEVDF scheduler later in the semester


jedi@tortuga: /proc/24050
jedi@tortuga:~/gitrepos/github/topsecret/cse536spring2024$ cat sched.sh
#!/bin/bash
pidof stress | sort -n | sed "s/ /\n/g" | while read p; do
#cat /proc/$p/cmdline
#echo ""
echo $p
cat /proc/$p/sched | grep vruntime
done
jedi@tortuga:~/gitrepos/github/topsecret/cse536spring2024$ watch ./sched.sh


jedi@tortuga: /proc/24050
jedi@tortuga:~$ stress -c 1 -i 1 -m 1
stress: info: [37444] dispatching hogs: 1 cpu, 1 io, 1 vm, 0 hdd

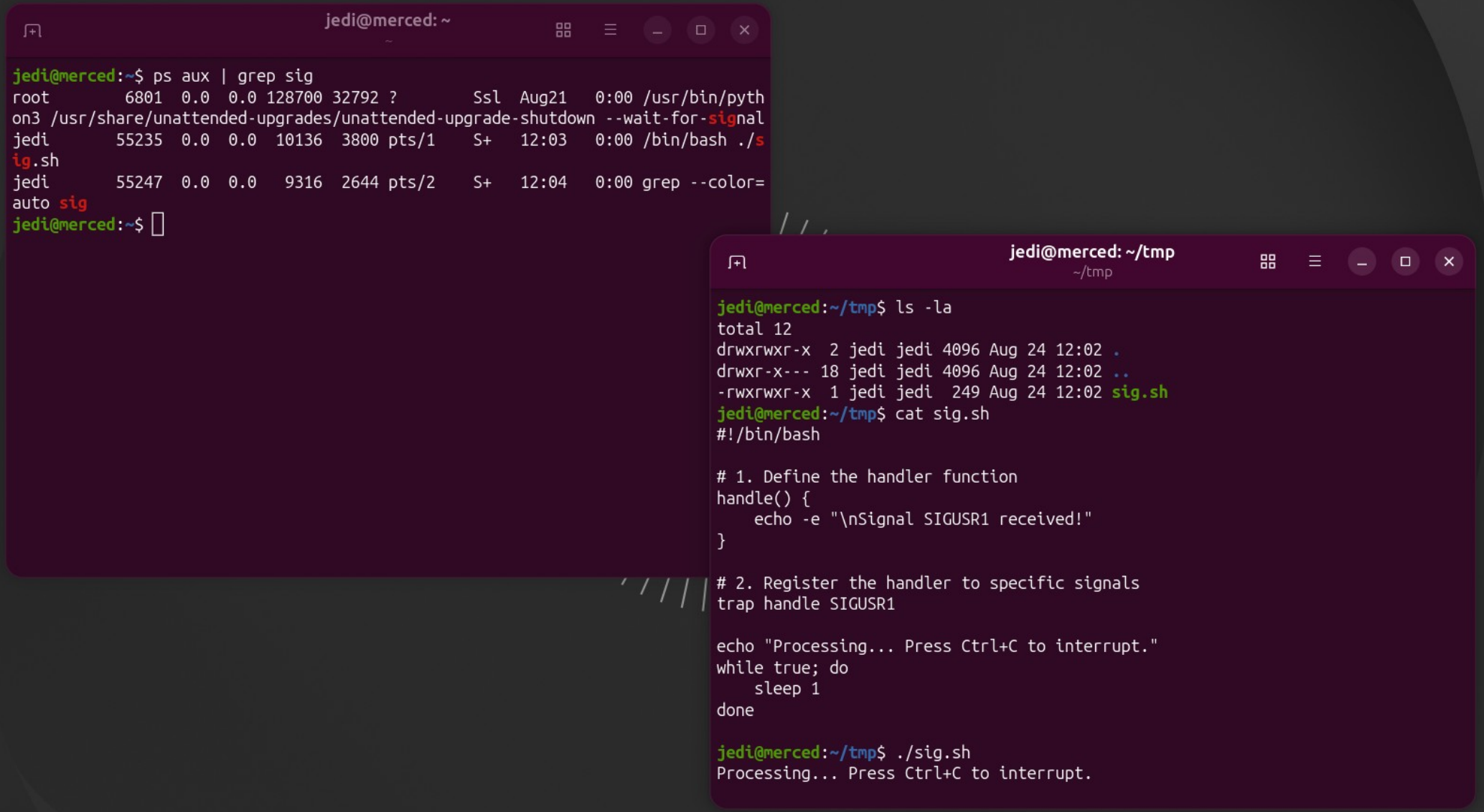

jedi@tortuga: /proc/24050
top - 13:31:51 up 1 day, 4:41, 4 users, load average: 1.47, 2.42, 3.21
Tasks: 440 total, 3 running, 437 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 7.1 sy, 6.9 ni, 79.7 id, 5.1 wa, 0.0 hi, 1.2 si, 0.0 st
MiB Mem : 31325.8 total, 17321.6 free, 8486.6 used, 5517.6 buff/cache
MiB Swap: 16384.0 total, 16384.0 free, 0.0 used. 20265.0 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
37445 jedi 26 6 3708 256 256 R 100.0 0.0 0:25.26 stress
37447 jedi 26 6 265856 97968 128 R 100.0 0.3 0:25.26 stress
37446 jedi 26 6 3708 128 128 D 19.5 0.0 0:04.92 stress
21808 jedi 26 6 1131.6g 211904 110448 S 2.0 0.7 0:40.38 chrome
2909 jedi 26 6 33.1g 318192 184720 S 1.7 1.0 32:07.17 chrome
9297 jedi 26 6 6149668 2.8g 2.7g S 1.7 9.3 11:36.08 Virtual+
15154 root 0 -20 0 0 0 I 1.3 0.0 0:01.95 kworker+
37260 root 0 -20 0 0 0 I 1.0 0.0 0:00.13 kworker+
1912 jedi -50 -15 143312 37800 8708 S 0.7 0.1 0:41.08 pipewir+
2863 jedi 26 6 33.3g 751168 559040 S 0.7 2.3 21:10.10 chrome
2910 jedi 26 6 32.5g 153956 106444 S 0.7 0.5 6:37.57 chrome
21791 jedi 26 6 1133.7g 345616 130920 S 0.7 1.1 3:03.42 chrome
29778 root 0 -20 0 0 0 I 0.7 0.0 0:02.21 kworker+
32688 root 0 -20 0 0 0 I 0.7 0.0 0:01.41 kworker+
37450 root 0 -20 0 0 0 I 0.7 0.0 0:00.18 kworker+
281 root -51 0 0 0 0 S 0.3 0.0 0:05.79 irq/86-+
1223 root 20 0 0 0 0 S 0.3 0.0 1:32.14 napi/ph+

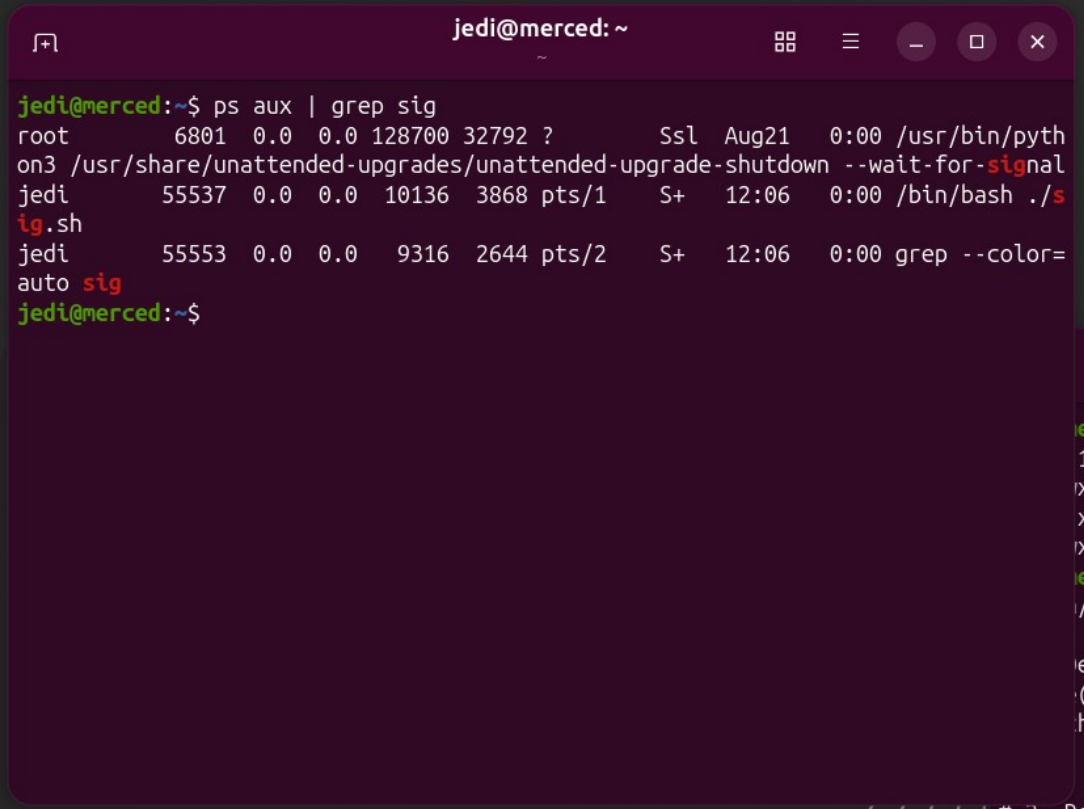

jedi@tortuga: /proc/24050
Every 2.0s: ./sched.sh tortuga: Fri Jan 26 13:34:03 2024
37447
se.vruntime : 2164454.807419
37446
se.vruntime : 2740210.094238
37445
se.vruntime : 4762379.104371
37444
se.vruntime : 4014571.764231

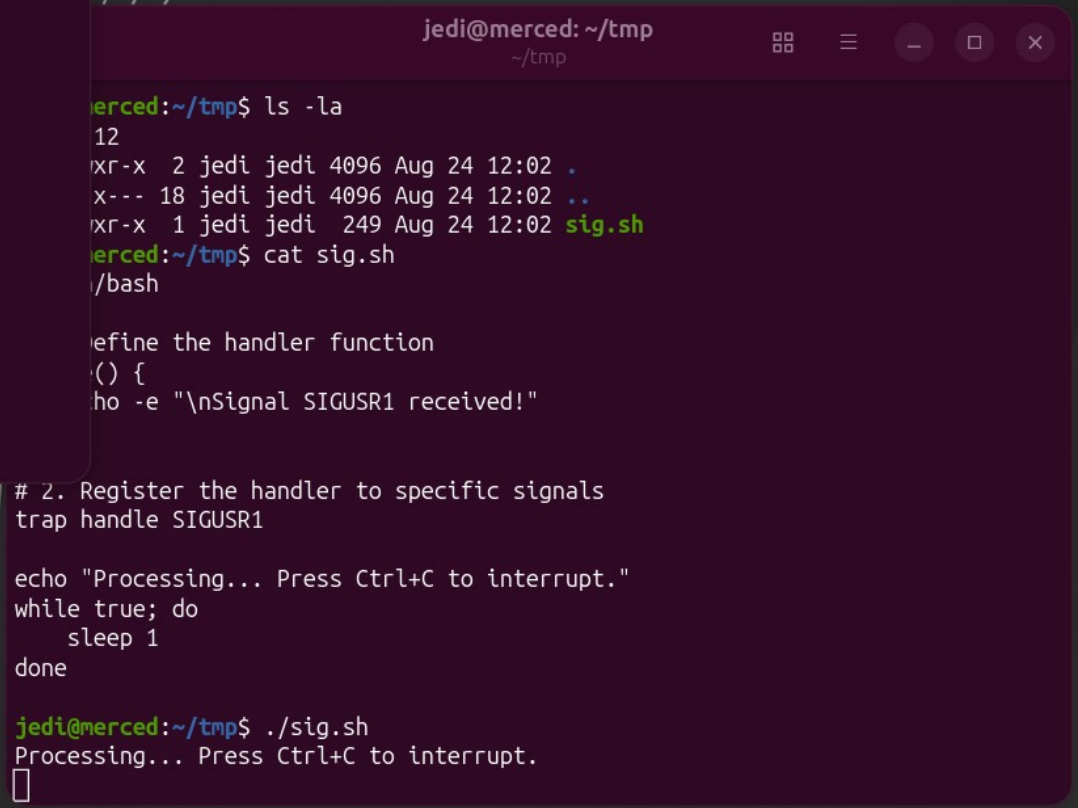

jedi@tortuga: /proc/24050
Every 2.0s: ./sched.sh tortuga: Fri Jan 26 13:34:51 2024
37447
se.vruntime : 3517317.756064
37446
se.vruntime : 3546412.653729
37445
se.vruntime : 3239799.395092
37444
se.vruntime : 4014571.764231

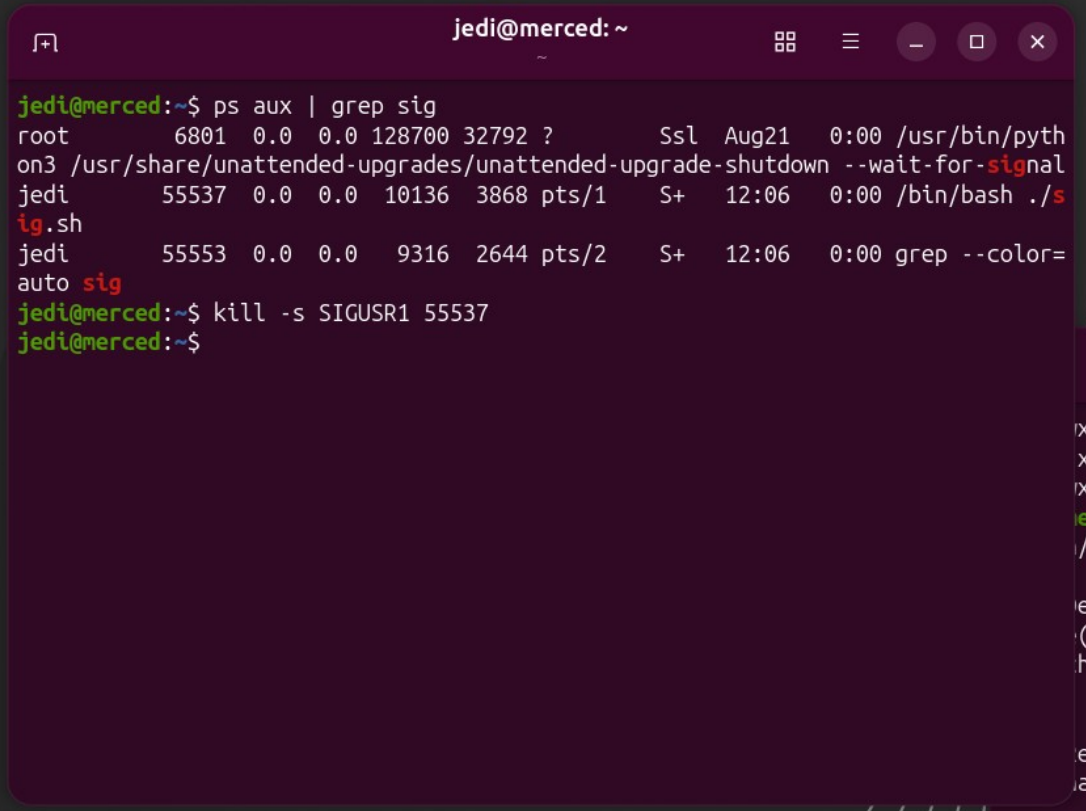
Demo

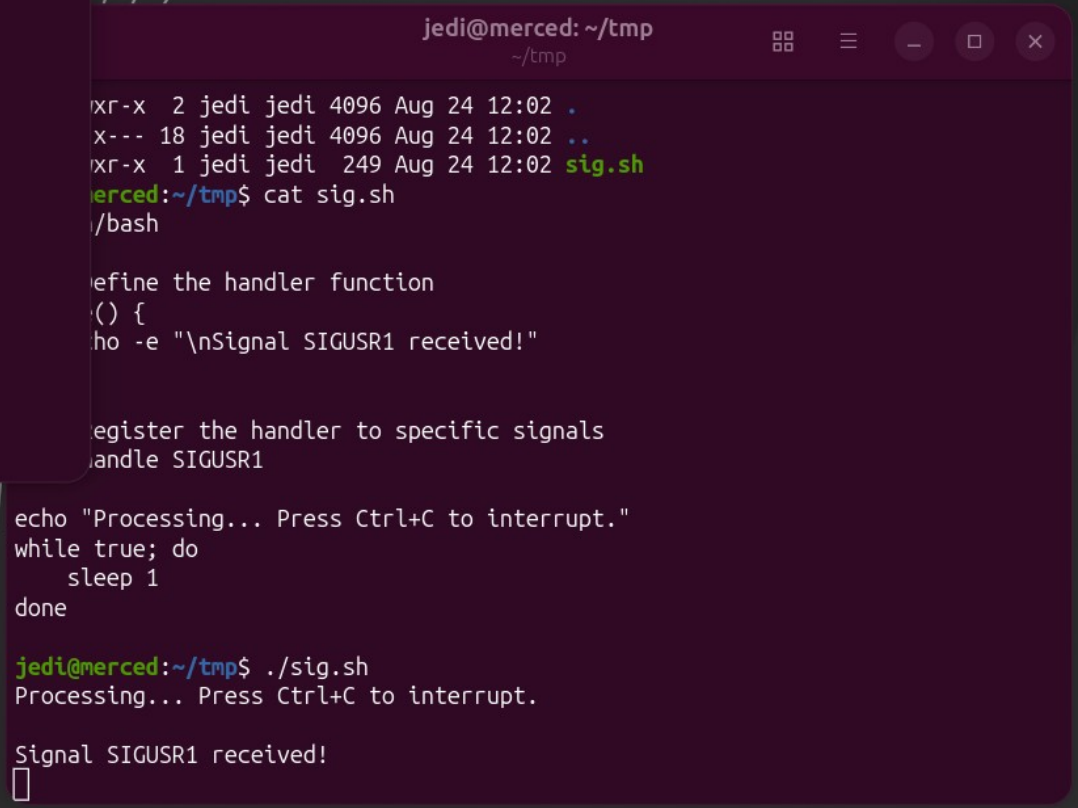
- Parent on bottom never changes
  - In a wait state
- CPU intensive (37445) stays pretty high all the time
- Memory intensive (37447) jumps back and forth
- I/O intensive (37446) usually the lowest

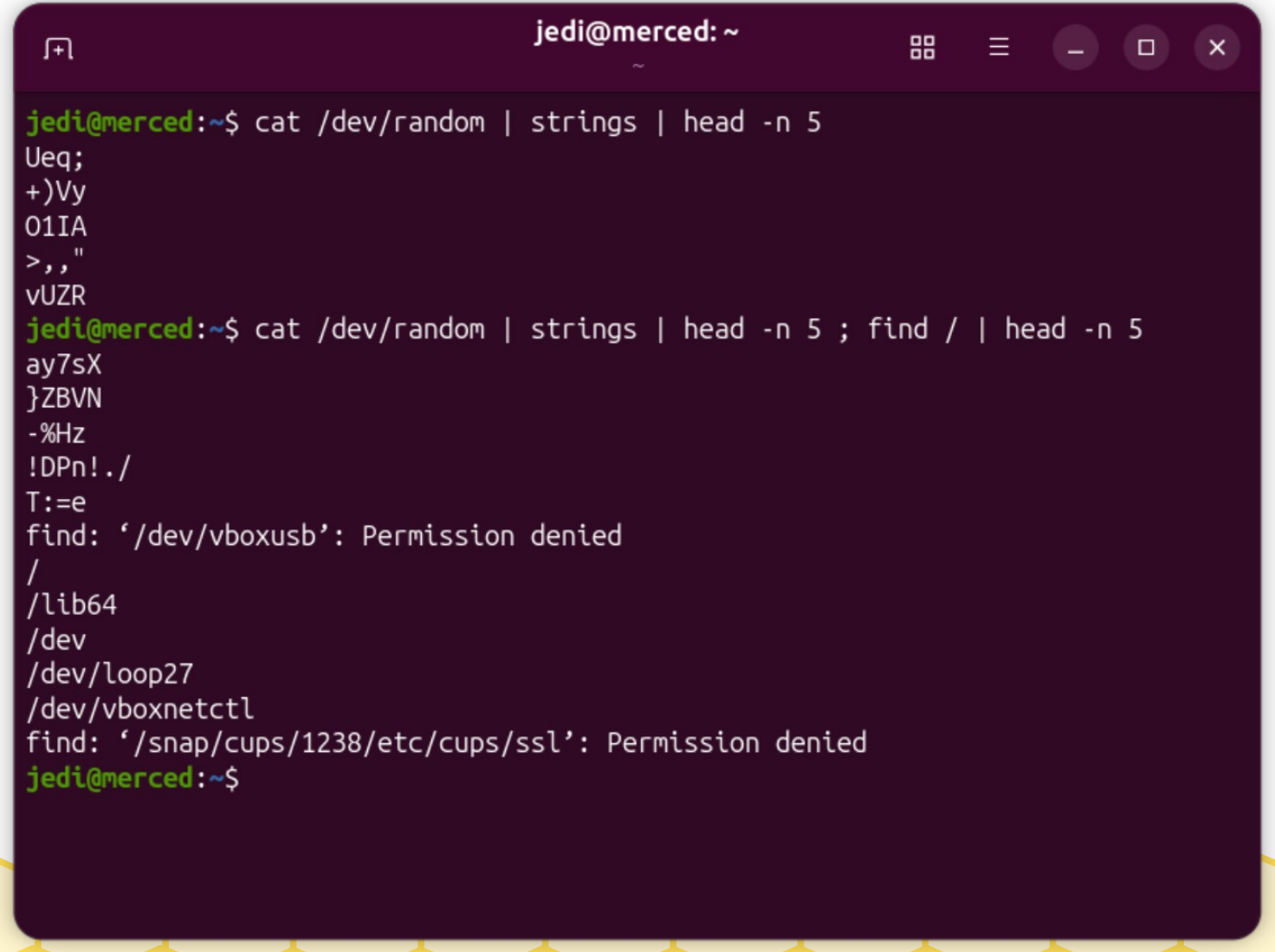

jedi@merced: ~
jedi@merced:~$ ps aux | grep sig
root 6801 0.0 0.0 128700 32792 ? Ssl Aug21 0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
jedi 55235 0.0 0.0 10136 3800 pts/1 S+ 12:03 0:00 /bin/bash ./sig.sh
jedi 55247 0.0 0.0 9316 2644 pts/2 S+ 12:04 0:00 grep --color=auto sig
jedi@merced:~$
jedi@merced: ~/tmp
~/tmp
jedi@merced:~/tmp$ ls -la
total 12
drwxrwxr-x 2 jedi jedi 4096 Aug 24 12:02 .
drwxr-x--- 18 jedi jedi 4096 Aug 24 12:02 ..
-rwxrwxr-x 1 jedi jedi 249 Aug 24 12:02 sig.sh
jedi@merced:~/tmp$ cat sig.sh
#!/bin/bash
# 1. Define the handler function
handle() {
echo -e "\nSignal SIGUSR1 received!"
}
# 2. Register the handler to specific signals
trap handle SIGUSR1
echo "Processing... Press Ctrl+C to interrupt."
while true; do
sleep 1
done
jedi@merced:~/tmp$ ./sig.sh
Processing... Press Ctrl+C to interrupt.

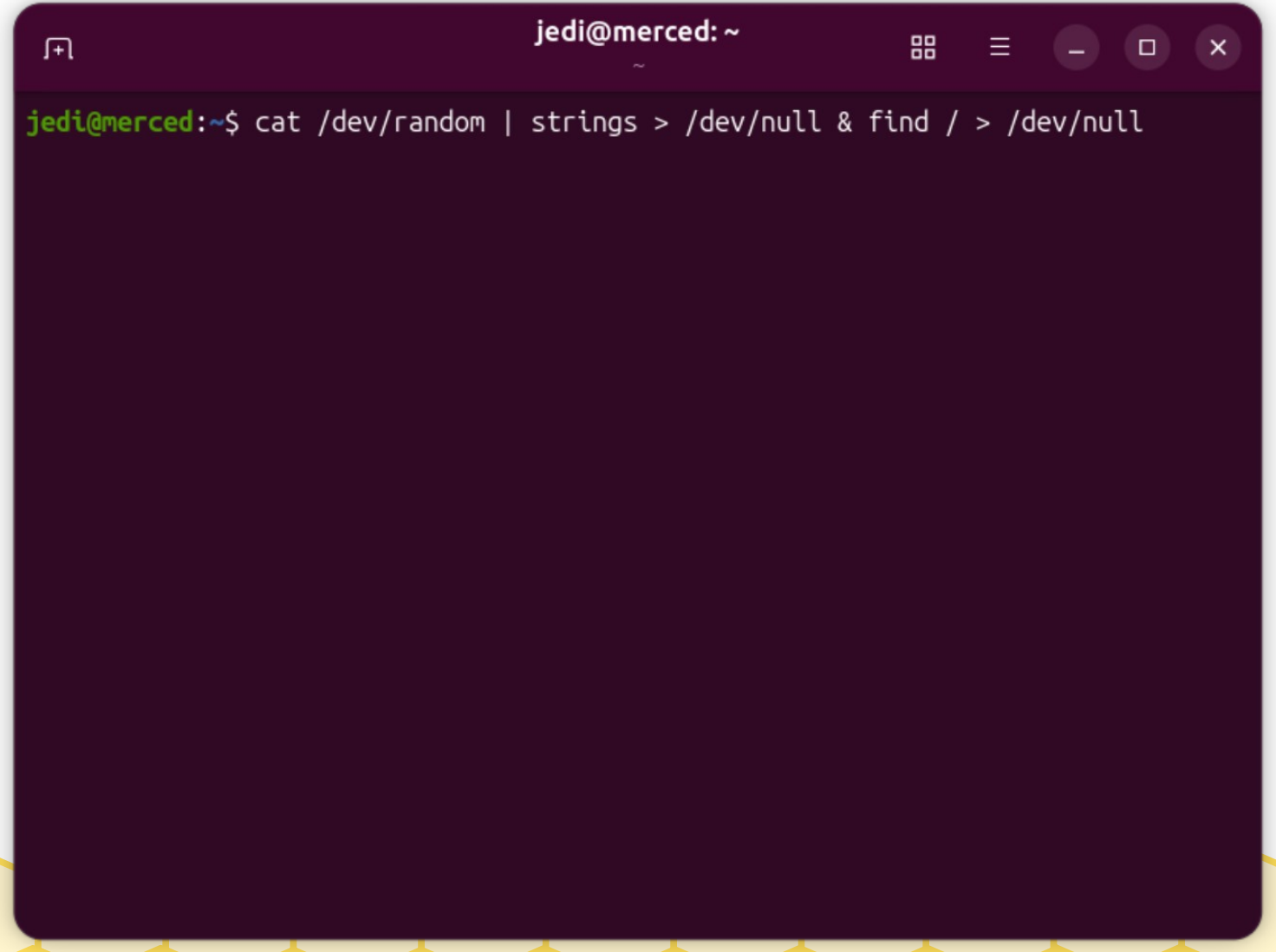

jedi@merced: ~
jedi@merced:~$ ps aux | grep sig
root 6801 0.0 0.0 128700 32792 ? Ssl Aug21 0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
jedi 55537 0.0 0.0 10136 3868 pts/1 S+ 12:06 0:00 /bin/bash ./sig.sh
jedi 55553 0.0 0.0 9316 2644 pts/2 S+ 12:06 0:00 grep --color=auto sig
jedi@merced:~$

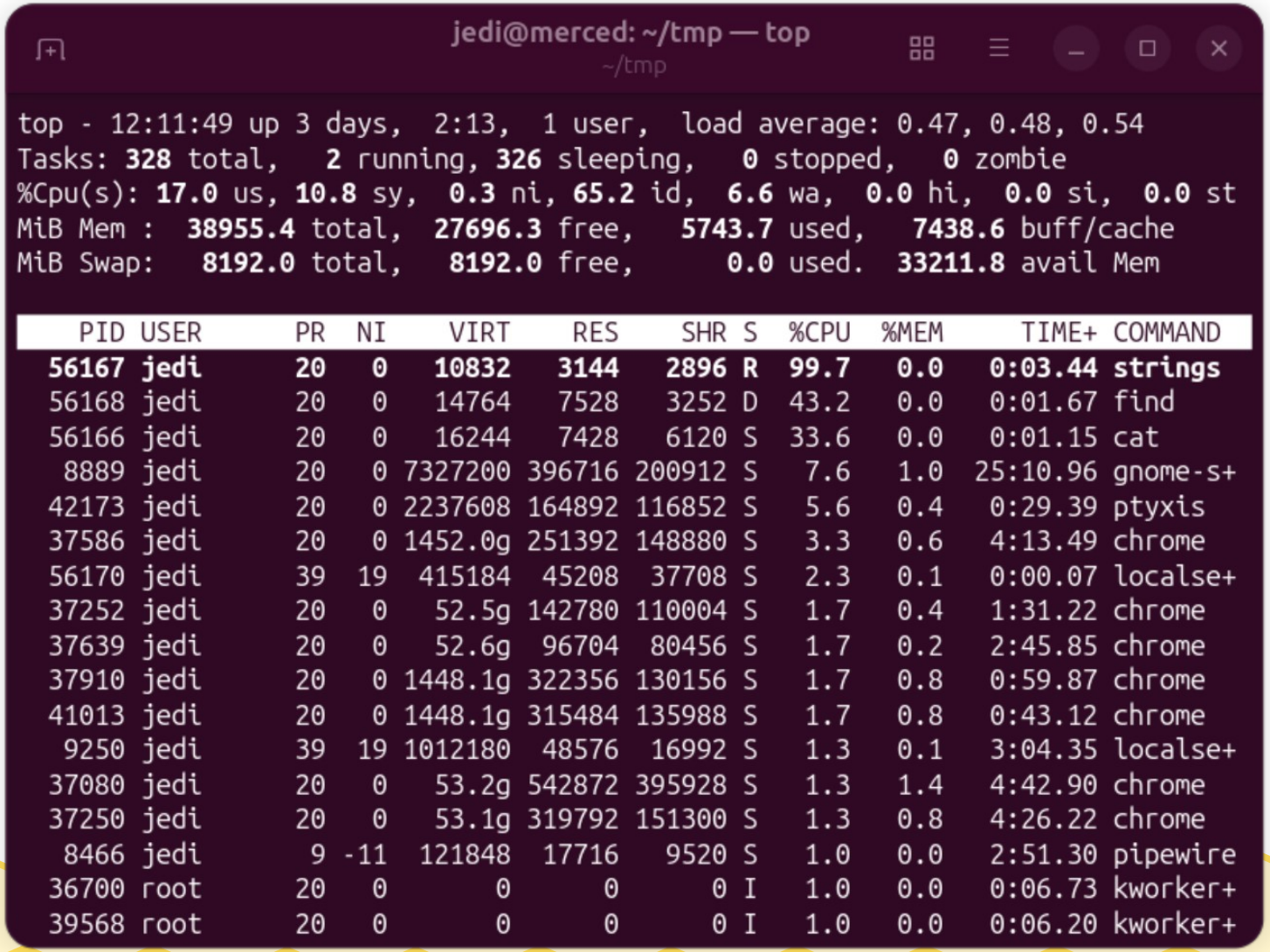

jedi@merced: ~/tmp
~/tmp
erced:~/tmp$ ls -la
12
xr-x 2 jedi jedi 4096 Aug 24 12:02 .
x--- 18 jedi jedi 4096 Aug 24 12:02 ..
xr-x 1 jedi jedi 249 Aug 24 12:02 sig.sh
erced:~/tmp$ cat sig.sh
/bash
efine the handler function
() {
ho -e "\nSignal SIGUSR1 received!"
2. Register the handler to specific signals
trap handle SIGUSR1
echo "Processing... Press Ctrl+C to interrupt."
while true; do
sleep 1
done
jedi@merced:~/tmp$ ./sig.sh
Processing... Press Ctrl+C to interrupt.

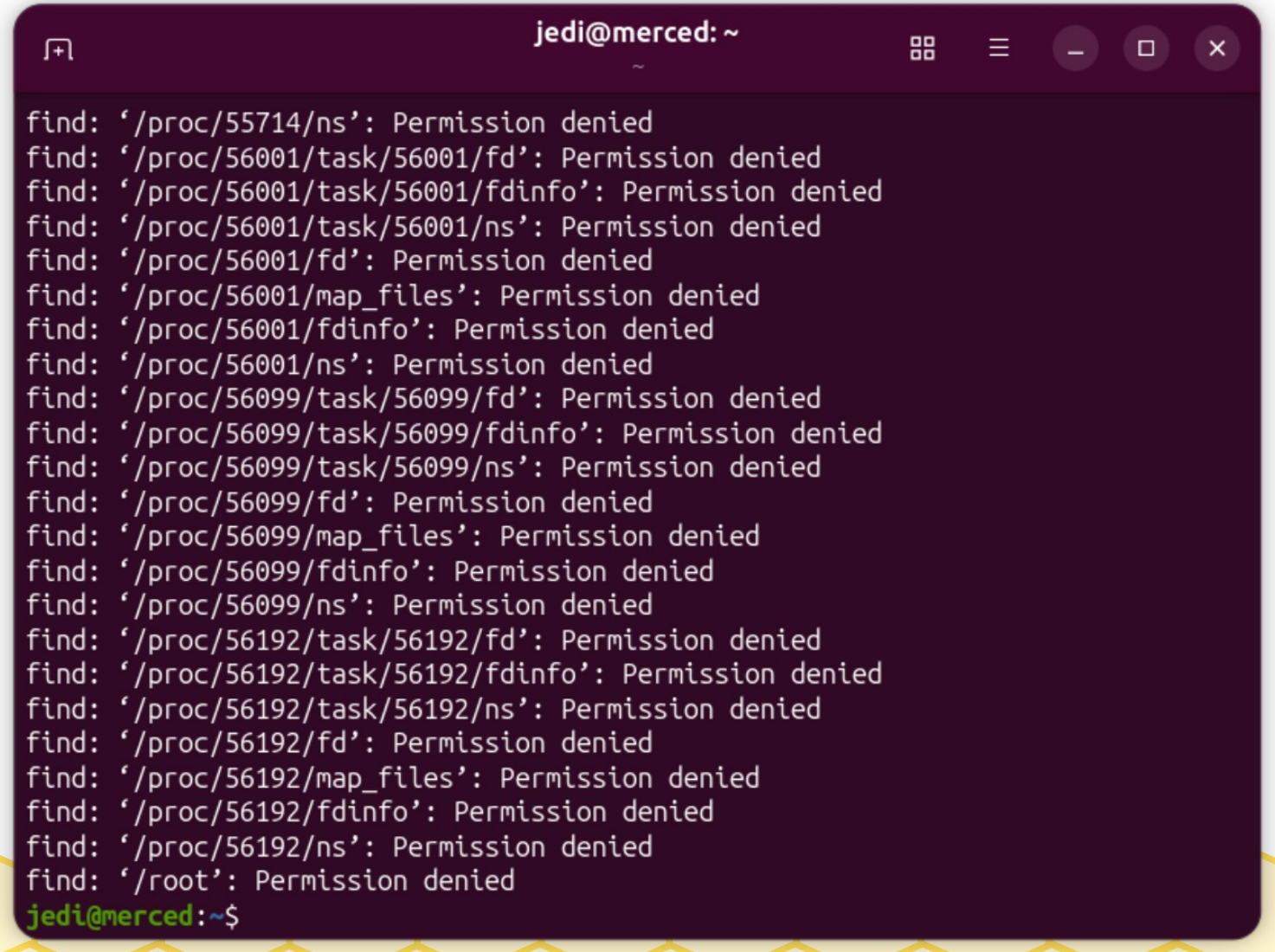

jedi@merced: ~
jedi@merced:~$ ps aux | grep sig
root 6801 0.0 0.0 128700 32792 ? Ssl Aug21 0:00 /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
jedi 55537 0.0 0.0 10136 3868 pts/1 S+ 12:06 0:00 /bin/bash ./sig.sh
jedi 55553 0.0 0.0 9316 2644 pts/2 S+ 12:06 0:00 grep --color=auto sig
jedi@merced:~$ kill -s SIGUSR1 55537
jedi@merced:~$


jedi@merced: ~/tmp
~/tmp
xr-x 2 jedi jedi 4096 Aug 24 12:02 .
x--- 18 jedi jedi 4096 Aug 24 12:02 ..
xr-x 1 jedi jedi 249 Aug 24 12:02 sig.sh
erced:~/tmp$ cat sig.sh
/bash
efine the handler function
() {
ho -e "\nSignal SIGUSR1 received!"
egister the handler to specific signals
andle SIGUSR1
echo "Processing... Press Ctrl+C to interrupt."
while true; do
sleep 1
done
jedi@merced:~/tmp$ ./sig.sh
Processing... Press Ctrl+C to interrupt.
Signal SIGUSR1 received!


jedi@merced: ~
jedi@merced:~$ cat /dev/random | strings | head -n 5
Ueq;
+)Vy
O1IA
>,,"
vUZR
jedi@merced:~$ cat /dev/random | strings | head -n 5 ; find / | head -n 5
ay7sX
}ZBVN
-%Hz
!DPn!./
T:=e
find: ‘/dev/vboxusb’: Permission denied
/
/lib64
/dev
/dev/loop27
/dev/vboxnetctl
find: ‘/snap/cups/1238/etc/cups/ssl’: Permission denied
jedi@merced:~$


jedi@merced: ~
jedi@merced:~$ cat /dev/random | strings > /dev/null & find / > /dev/null


jedi@merced: ~/tmp — top
~/tmp
top - 12:11:49 up 3 days, 2:13, 1 user, load average: 0.47, 0.48, 0.54
Tasks: 328 total, 2 running, 326 sleeping, 0 stopped, 0 zombie
%Cpu(s): 17.0 us, 10.8 sy, 0.3 ni, 65.2 id, 6.6 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 38955.4 total, 27696.3 free, 5743.7 used, 7438.6 buff/cache
MiB Swap: 8192.0 total, 8192.0 free, 0.0 used. 33211.8 avail Mem
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
56167 jedi 20 0 10832 3144 2896 R 99.7 0.0 0:03.44 strings
56168 jedi 20 0 14764 7528 3252 D 43.2 0.0 0:01.67 find
56166 jedi 20 0 16244 7428 6120 S 33.6 0.0 0:01.15 cat
8889 jedi 20 0 7327200 396716 200912 S 7.6 1.0 25:10.96 gnome-s+
42173 jedi 20 0 2237608 164892 116852 S 5.6 0.4 0:29.39 ptyxis
37586 jedi 20 0 1452.0g 251392 148880 S 3.3 0.6 4:13.49 chrome
56170 jedi 39 19 415184 45208 37708 S 2.3 0.1 0:00.07 localse+
37252 jedi 20 0 52.5g 142780 110004 S 1.7 0.4 1:31.22 chrome
37639 jedi 20 0 52.6g 96704 80456 S 1.7 0.2 2:45.85 chrome
37910 jedi 20 0 1448.1g 322356 130156 S 1.7 0.8 0:59.87 chrome
41013 jedi 20 0 1448.1g 315484 135988 S 1.7 0.8 0:43.12 chrome
9250 jedi 39 19 1012180 48576 16992 S 1.3 0.1 3:04.35 localse+
37080 jedi 20 0 53.2g 542872 395928 S 1.3 1.4 4:42.90 chrome
37250 jedi 20 0 53.1g 319792 151300 S 1.3 0.8 4:26.22 chrome
8466 jedi 9 -11 121848 17716 9520 S 1.0 0.0 2:51.30 pipewire
36700 root 20 0 0 0 0 I 1.0 0.0 0:06.73 kworker+
39568 root 20 0 0 0 0 I 1.0 0.0 0:06.20 kworker+


jedi@merced: ~
find: ‘/proc/55714/ns’: Permission denied
find: ‘/proc/56001/task/56001/fd’: Permission denied
find: ‘/proc/56001/task/56001/fdinfo’: Permission denied
find: ‘/proc/56001/task/56001/ns’: Permission denied
find: ‘/proc/56001/fd’: Permission denied
find: ‘/proc/56001/map_files’: Permission denied
find: ‘/proc/56001/fdinfo’: Permission denied
find: ‘/proc/56001/ns’: Permission denied
find: ‘/proc/56099/task/56099/fd’: Permission denied
find: ‘/proc/56099/task/56099/fdinfo’: Permission denied
find: ‘/proc/56099/task/56099/ns’: Permission denied
find: ‘/proc/56099/fd’: Permission denied
find: ‘/proc/56099/map_files’: Permission denied
find: ‘/proc/56099/fdinfo’: Permission denied
find: ‘/proc/56099/ns’: Permission denied
find: ‘/proc/56192/task/56192/fd’: Permission denied
find: ‘/proc/56192/task/56192/fdinfo’: Permission denied
find: ‘/proc/56192/task/56192/ns’: Permission denied
find: ‘/proc/56192/fd’: Permission denied
find: ‘/proc/56192/map_files’: Permission denied
find: ‘/proc/56192/fdinfo’: Permission denied
find: ‘/proc/56192/ns’: Permission denied
find: ‘/root’: Permission denied
jedi@merced:~$

UNIX

- Everything is a process or a file
  - Process hierarchy
    - File descriptor tables
  - File hierarchy
- Pipes and other interprocess communication
- Concurrency
  - Signals
  - Wait queues
  - `fork`() and `exec`()