1) /etc has system configs attackers may alter, **/bin** and **/usr** hold essential binaries that could be replaced, **/var** keeps logs showing intrusions, while **/tmp**, **/opt, /boot**, and **/home** may be abused for payloads, third-party tools, boot persistence, or user backdoors.

/etc — Holds system-wide configuration files (e.g., `sshd_config`, `passwd`, `sudoers`); attackers modify these to create backdoors or change authentication.

/bin — Contains essential user-level binaries used at boot/single-user mode (e.g., `ls`, `bash`); replacing these with trojans gives broad control.

/usr — Contains the bulk of applications, binaries and libraries (`/usr/bin`, `/usr/lib`); compromised files here affect many services.

/var — Stores variable data including logs (`/var/log`), mail spools and web content; logs are primary evidence of intrusion.

/tmp — World-writable temp directory frequently abused to drop and run attacker tools or payloads.

/opt — For optional/third-party installs; attackers may place standalone tools here to avoid package managers.

/boot — Holds kernels, initramfs and bootloader config (e.g., GRUB); tampering enables deep, persistent (boot-time) compromises.

/home — User data, dotfiles and `.ssh` keys; attackers often add keys or scripts here for persistence.

2)**$ mkdir -p ~/projects/{client_work/{web/{frontend,backend,database},mobile/{ios,android}},personal/{experiments,archive},shared/{templates,resources}}**

```
Herve1@Herve MINGW64 ~
$ find ~/projects -type d | sed 's|^|/|'
//c/Users/Herve1/projects
//c/Users/Herve1/projects/client_work
//c/Users/Herve1/projects/client_work/mobile
//c/Users/Herve1/projects/client_work/mobile/android
//c/Users/Herve1/projects/client_work/mobile/ios
//c/Users/Herve1/projects/client_work/web
//c/Users/Herve1/projects/client_work/web/backend
//c/Users/Herve1/projects/client_work/web/database
//c/Users/Herve1/projects/client_work/web/frontend
//c/Users/Herve1/projects/personal
//c/Users/Herve1/projects/personal/archive
//c/Users/Herve1/projects/personal/experiments
//c/Users/Herve1/projects/shared
//c/Users/Herve1/projects/shared/resources
//c/Users/Herve1/projects/shared/templates
```

3)cd ../../../personal/experiments

cd ../../shared/templates

cd ../../client_work/web/frontend

```
Herve1@Herve MINGW64 ~/projects/client_work/web/frontend
$ cd ../../../personal/experiments

Herve1@Herve MINGW64 ~/projects/personal/experiments
$ cd ../../shared/templates

Herve1@Herve MINGW64 ~/projects/shared/templates
$ cd ../../client_work/web/frontend

Herve1@Herve MINGW64 ~/projects/client_work/web/frontend
$
```

4)Create project root and subdirectories

Create 15 HTML files

touch web_project/html/{index,about,contact}.html
web_project/html/page_{001..012}.html

```
Herve1@Herve MINGW64 ~
$ mkdir -p web_project/{html,css,js,backups}

Herve1@Herve MINGW64 ~
$ touch web_project/html/{index,about,contact}.html web
_project/html/page_{001..012}.html

Herve1@Herve MINGW64 ~
$ cd web_project/html

Herve1@Herve MINGW64 ~/web_project/html
$ ls
about.html     page_003.html  page_008.html
contact.html   page_004.html  page_009.html
index.html     page_005.html  page_010.html
page_001.html  page_006.html  page_011.html
page_002.html  page_007.html  page_012.html

Herve1@Herve MINGW64 ~/web_project/html
$
```

Create 8 CSS files

touch web_project/css/{main,reset,theme_light,theme_dark,mobile,tablet,desktop,print}.css

```
Herve1@Herve MINGW64 ~
$ touch web_project/css/{main,reset,theme_light,theme_d
ark,mobile,tablet,desktop,print}.css

Herve1@Herve MINGW64 ~
$ cd web_project/css

Herve1@Herve MINGW64 ~/web_project/css
$ ls
desktop.css  mobile.css  reset.css   theme_dark.css
main.css     print.css   tablet.css  theme_light.css

Herve1@Herve MINGW64 ~/web_project/css
$
```

Create 6 JavaScript files
touch web_project/js/{main_script,extra_script,helper_util,dom_util,app_config,db_config}.js

```
Herve1@Herve MINGW64 ~
$ touch web_project/js/{main_script,extra_script,helper
_util,dom_util,app_config,db_config}.js

Herve1@Herve MINGW64 ~
$ cd web_project/js

Herve1@Herve MINGW64 ~/web_project/js
$ ls
app_config.js  dom_util.js      helper_util.js
db_config.js   extra_script.js  main_script.js

Herve1@Herve MINGW64 ~/web_project/js
$
```

Create 20 backup files

for l in a b c d; do
** touch ${l}.{bak,old,tmp,backup,save}**
done

```
Herve1@Herve MINGW64 ~/web_project/backups
$ for l in a b c d; do
  touch ${l}.{bak,old,tmp,backup,save}
done

Herve1@Herve MINGW64 ~/web_project/backups
$ ls
a.backup  a.tmp     b.save    c.old     d.bak
a.bak     b.backup  b.tmp     c.save    d.old
a.old     b.bak     c.backup  c.tmp     d.save
a.save    b.old     c.bak     d.backup  d.tmp

Herve1@Herve MINGW64 ~/web_project/backups
$
```

5)Move all files ending in numbers to **archive/**

```
Herve1@Herve MINGW64 ~
$ mv *[0-9].* archive/
```

Copy all CSS files except those containing **"mobile"** or **"tablet" to desktop/**

```
Herve1@Herve MINGW64 ~
$ cp !( *mobile*|*tablet* ).css desktop/
```

List only files with exactly 3 characters before the dot

```
Herve1@Herve MINGW64 ~/wildcard_demo
$ ls ???.*
abc.txt  car.js  dog.sh  run.rb  zoo.py
```

Find files that start with any consonant (not a vowel)

```
Herve1@Herve MINGW64 ~/wildcard_demo
$ ls [b-df-hj-np-tv-zB-DF-HJ-NP-TV-Z]*
car.js        dog.sh   main.css    page_001.html  read.me    tablet.css       zoo.py
contact.html  file.tx  mobile.css  page_002.html  reset.css  theme_dark.css
desktop.css   go.py    my.cn       page_003.html  run.rb     theme_light.css
```

Identify files where the extension has exactly 2 characters

```
Herve1@Herve MINGW64 ~/wildcard_demo
$ ls *.[a-zA-Z][a-zA-Z]
car.js  dog.sh  file.tx  go.py  my.cn  read.me  run.rb  zoo.py
```

6)What this creates

log_2024-01-01.txt … log_2024-03-31.txt (all valid days of Jan–Mar 2024).

config_development_web.conf … config_production_db.conf (9 files total).

test_A_10_input.txt … test_C_12_output.txt (18 files total).

```
Herve1@Herve MINGW64 ~
$ for month in {01..03}; do
  for day in {01..31}; do
    # Only create if valid date
    if date -d "2024-$month-$day" >/dev/null 2>&1; then
      touch log_2024-$month-$day.txt
    fi
  done
done

Herve1@Herve MINGW64 ~
$ touch config_{development,staging,production}_{web,api,db}.conf

Herve1@Herve MINGW64 ~
$ touch test_{A..C}_{10..12}_{input,output}.txt
```

```
config_development_api.conf
config_development_db.conf
config_development_web.conf
config_production_api.conf
config_production_db.conf
config_production_web.conf
config_staging_api.conf
config_staging_db.conf
config_staging_web.conf
d2.old
d3.tmp
d4.backup
d5.save
log_2024-01-01.txt
log_2024-01-02.txt
log_2024-01-03.txt
log_2024-01-04.txt
log_2024-01-05.txt
log_2024-01-06.txt
log_2024-01-07.txt
log_2024-01-08.txt
log_2024-01-09.txt
log_2024-01-10.txt
log_2024-01-11.txt
log_2024-01-12.txt
log_2024-01-13.txt
log_2024-01-14.txt
log_2024-01-15.txt
log_2024-01-16.txt
log_2024-01-17.txt
log_2024-01-18.txt
log_2024-01-19.txt
log_2024-01-20.txt
log_2024-01-21.txt
```

7)I created two configuration files with the same text: one using Linux line endings (LF) and one using Windows line endings (CRLF). Comparing them with diff showed every line as different, cmp revealed the raw byte differences (\r characters), and comm treated matching lines as distinct. This demonstrates that invisible line-ending formats can cause cross-platform compatibility issues. It highlights the need to normalize line endings (e.g., with dos2unix or Git's core.autocrlf) to avoid subtle errors when sharing files between Linux and Windows

```
Herve1@Herve MINGW64 ~
$ echo -e "config=value\nsetting=on\nmode=prod" > linux.conf

Herve1@Herve MINGW64 ~
$ unix2dos linux.conf windows.conf
unix2dos: converting file linux.conf to DOS format...
unix2dos: converting file windows.conf to DOS format...

Herve1@Herve MINGW64 ~
$
```

Compare with tools

Prints byte-by-byte differences: shows the extra \r (carriage return) before \n

```
Herve1@Herve MINGW64 ~
$ comm linux.conf windows.conf
                config=value
                setting=on
                mode=prod
```

8)Creating Test Environment

```
ubuntu1@Ubuntu1:~$ mkdir -p audit_env/{docs,logs,tmp,hidden,empty}
ubuntu1@Ubuntu1:~$ cd audit_env
ubuntu1@Ubuntu1:~/audit_env$ echo "Report" > docs/report.txt
ubuntu1@Ubuntu1:~/audit_env$ head -c 1K </dev/urandom > docs/large
1.bin
ubuntu1@Ubuntu1:~/audit_env$ head -c 2K </dev/urandom > docs/large
2.bin
ubuntu1@Ubuntu1:~/audit_env$ touch -t 202409010101 logs/old.log
   # very old
ubuntu1@Ubuntu1:~/audit_env$ touch -t 202509250101 logs/recent.log
   # ~72h ago
ubuntu1@Ubuntu1:~/audit_env$ touch -t 202509270101 logs/today.log
   # within 24h
```

Using the Find command to find files that are larger than the average size

```
ubuntu1@Ubuntu1:~/audit_env$ avg=$(find . -type f -printf "%s\n" |
 awk '{sum+=$1; n++} END{if(n>0) print int(sum/n)}')
ubuntu1@Ubuntu1:~/audit_env$ find . -type f -size +"${avg}c"
./docs/large1.bin
./docs/large2.bin
```

Files that were modified within last 72h but not last 24h

```
ubuntu1@Ubuntu1:~/audit_env$ find . -type f -mtime -3 ! -mtime -1
./logs/today.log
ubuntu1@Ubuntu1:~/audit_env$
```

Empty Directories or Containing only hidden files

```
ubuntu1@Ubuntu1:~/audit_env$ find . -type d -empty
find . -type d ! -empty -exec sh -c '
  for d; do
    files=$(ls -A "$d")
    [[ -n "$files" && "$files" == .* ]] && echo "$d"
  done
' sh {} +
./empty/dir
sh: 4: [[: not found
sh: 4: [[: not found
sh: 4: [[: not found
sh: 4: [[: not found
sh: 4: [[: not found
```

World-writable files

```
ubuntu1@Ubuntu1:~$ find . -type f -perm -002
./Desktop/prac/games.txt
./doc.txt
./audit_env/docs/open.txt
ubuntu1@Ubuntu1:~$
```

Owned by users rather than me or root

```
ubuntu1@Ubuntu1:~$ find . -type f ! -user $USER ! -user root
./audit_env/docs/otheruser_testfile.txt
ubuntu1@Ubuntu1:~$
```

Temporaly/backup files (*.tmp, *~, *.bak)

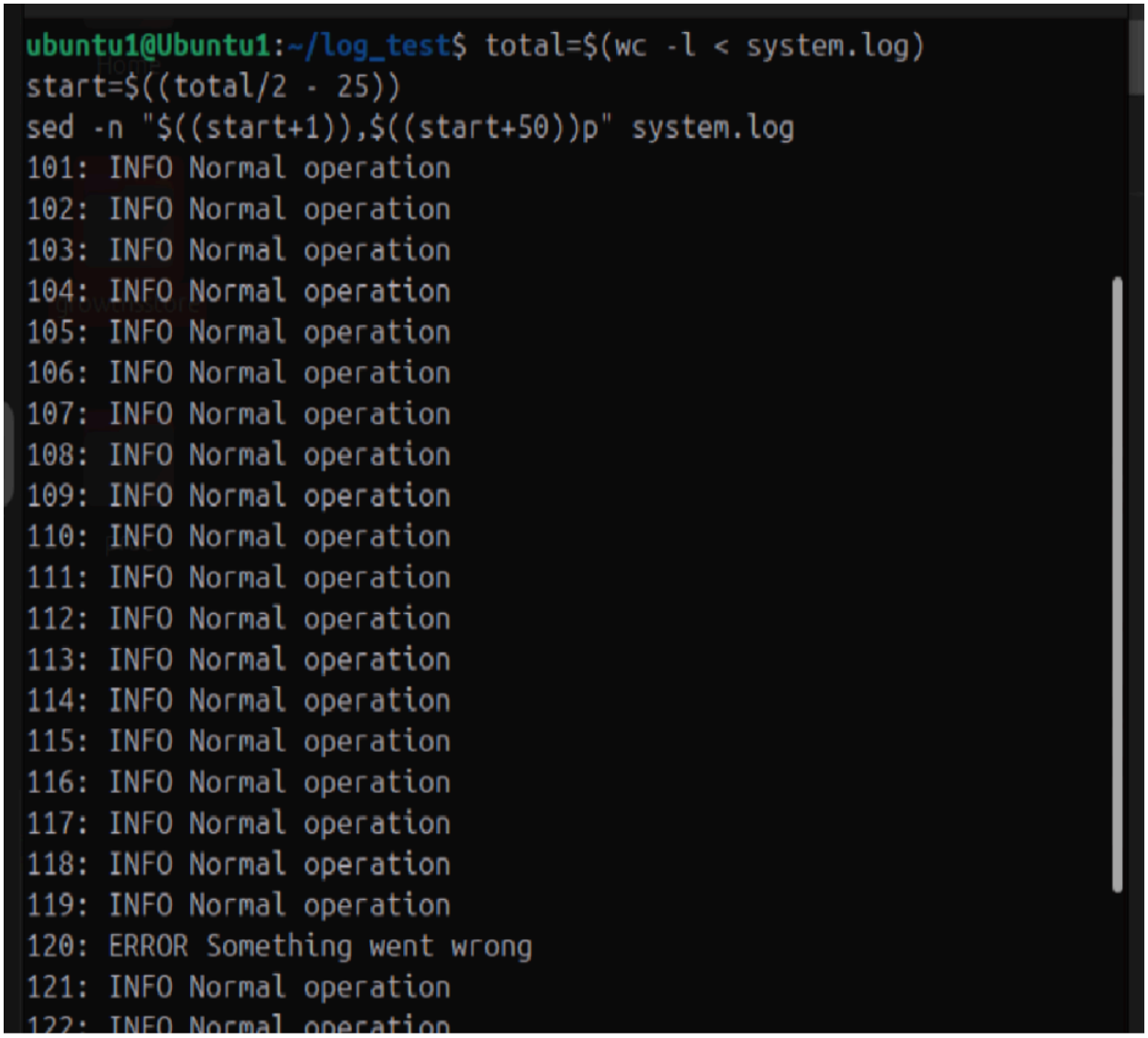
```
ubuntu1@Ubuntu1:~$ find . -type f \( -name '*.tmp' -o -name '*~' -
o -name '*.bak' \)
./audit_env/tmp/file.tmp
./audit_env/tmp/notes.bak
./audit_env/tmp/data~
```

9) Creating a large log file

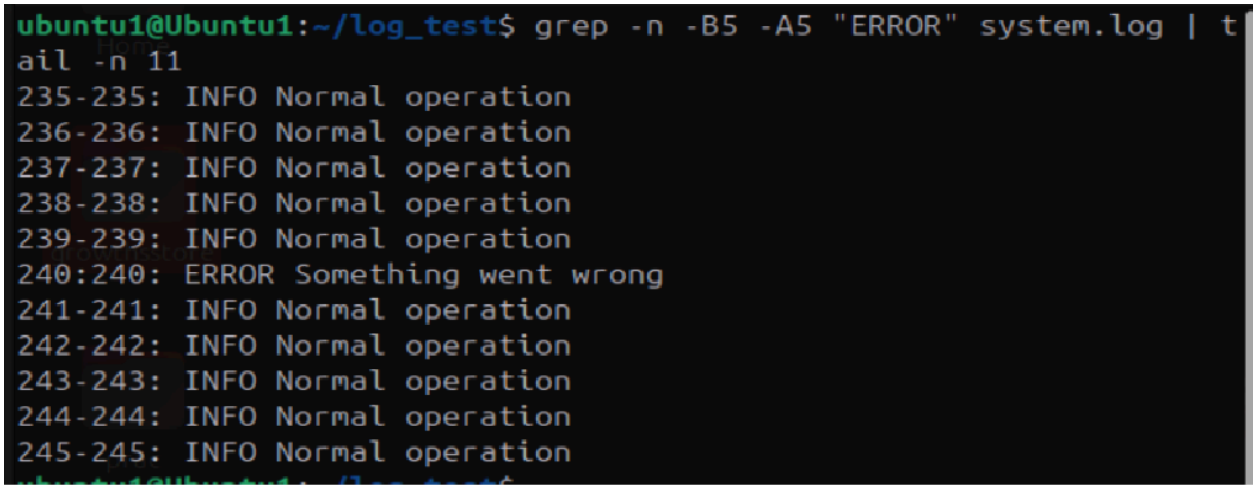
create one with 200+ lines

```
ubuntu1@Ubuntu1:~$ mkdir -p log_test
ubuntu1@Ubuntu1:~$ cd log_test
ubuntu1@Ubuntu1:~/log_test$ for i in {1..250}; do
  if (( i % 20 == 0 )); then
    echo "$i: ERROR Something went wrong" >> system.log
  else
    echo "$i: INFO Normal operation" >> system.log
  fi
done
```

Displaying 50Lines

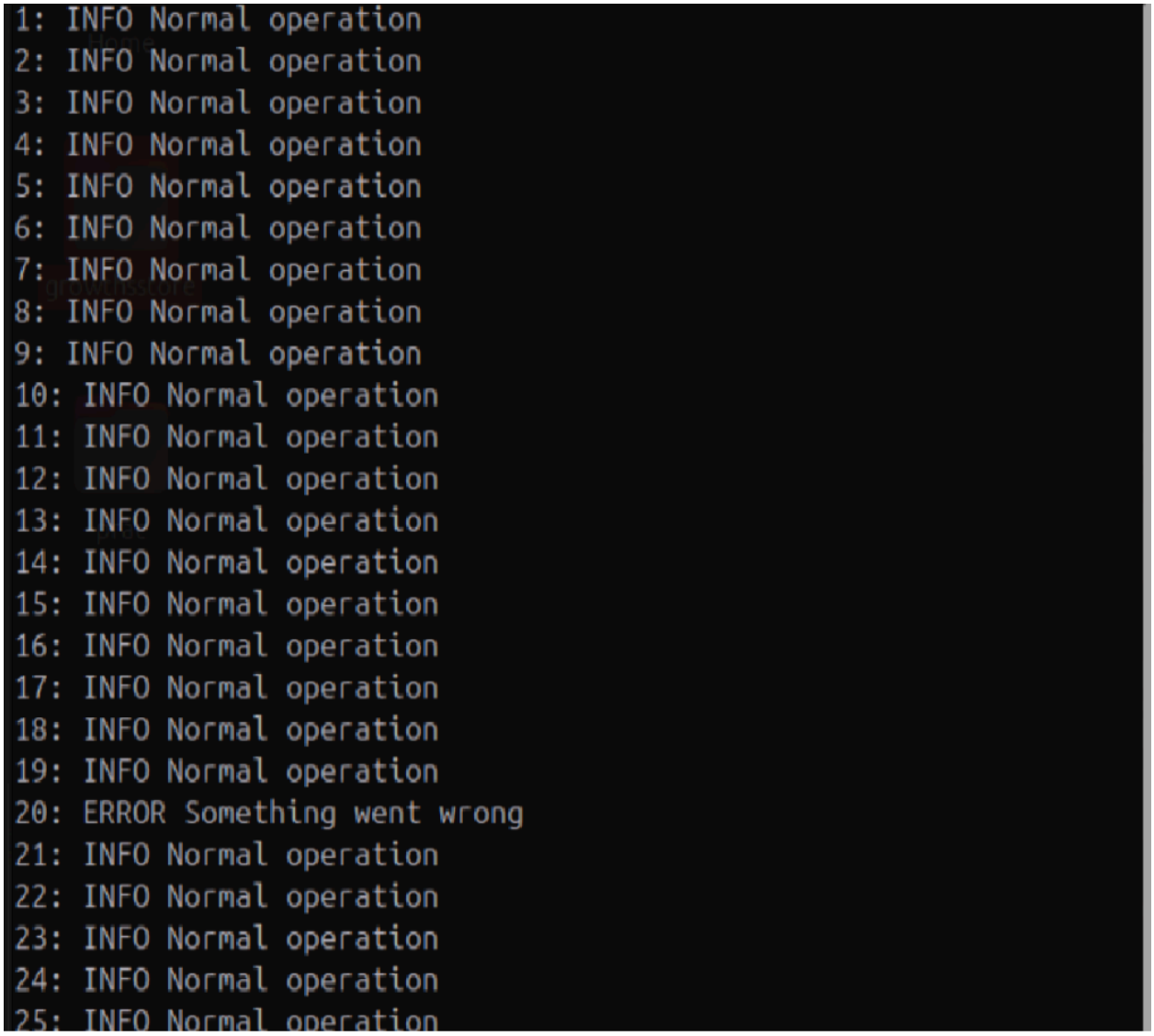


Finding the last occurrence of a word (ERROR) with 5 lines of context



Comparing the efficiency of viewing with different tools, “less” and “cat”

less is more efficient over SSH → it fetches and displays pages, unlike cat which sends the full file at once. cat streams the entire file immediately → can be slow on large files, consumes bandwidth. less displays one screen at a time, fetches data as needed → saves bandwidth, easier to scroll/search



Extracting only lines with error patterns, preserving line numbers

```
ubuntu1@Ubuntu1:~/log_test$ grep -n "ERROR" system.log
20:20: ERROR Something went wrong
40:40: ERROR Something went wrong
60:60: ERROR Something went wrong
80:80: ERROR Something went wrong
100:100: ERROR Something went wrong
120:120: ERROR Something went wrong
140:140: ERROR Something went wrong
160:160: ERROR Something went wrong
180:180: ERROR Something went wrong
200:200: ERROR Something went wrong
220:220: ERROR Something went wrong
240:240: ERROR Something went wrong
```

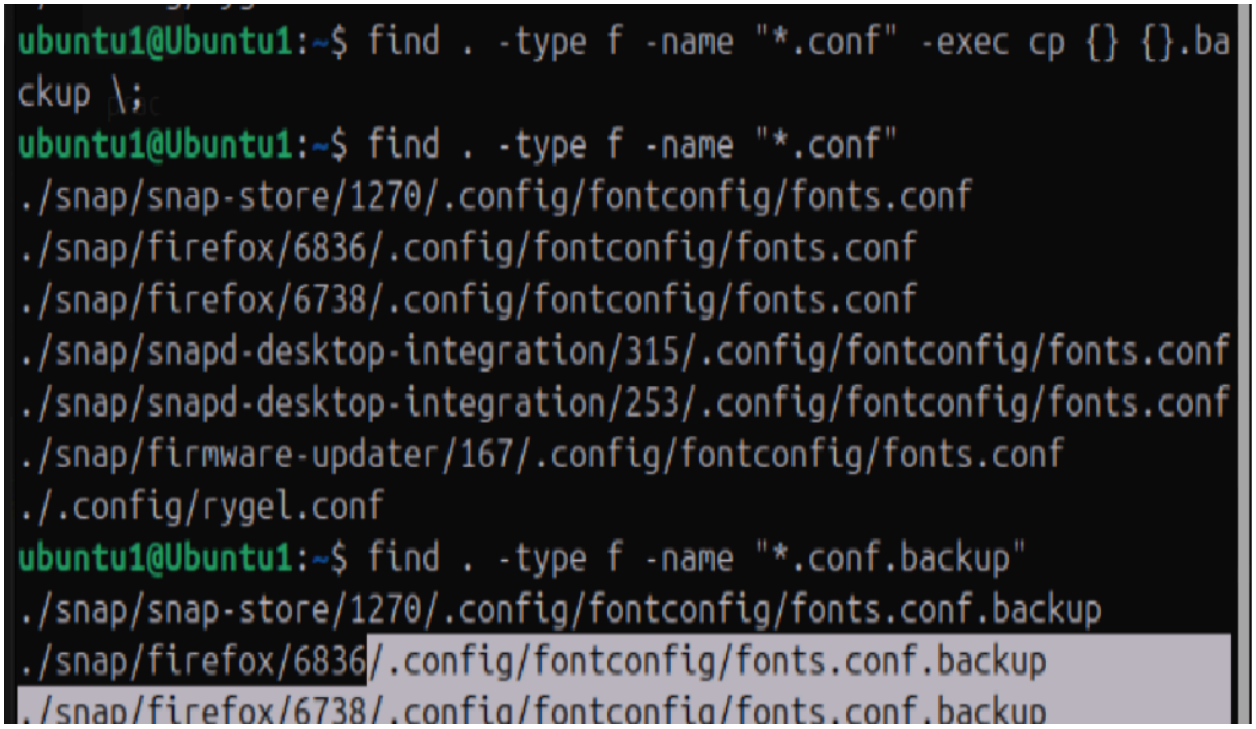
10)change permissions to 644 for all files except executables

```
ubuntu1@Ubuntu1:~/audit_env$ sudo find . -type f ! -perm 755 -exec
 chmod 644 {} \;
ubuntu1@Ubuntu1:~/audit_env$ ls -l
total 20
drwxrwxr-x 2 ubuntu1 ubuntu1 4096 Sep 28 17:25 docs
drwxrwxr-x 3 ubuntu1 ubuntu1 4096 Sep 28 17:06 empty
drwxrwxr-x 2 ubuntu1 ubuntu1 4096 Sep 28 17:06 hidden
drwxrwxr-x 2 ubuntu1 ubuntu1 4096 Sep 28 17:05 logs
drwxrwxr-x 2 ubuntu1 ubuntu1 4096 Sep 28 17:06 tmp
```

Calculate total disk space used by files older than 30 days

```
ubuntu1@Ubuntu1:~$ find . -type f -mtime +30 -exec du -ch {} + | t
ail -n 1
15M     total
ubuntu1@Ubuntu1:~$
```

Backingup all configuration files (*.conf)

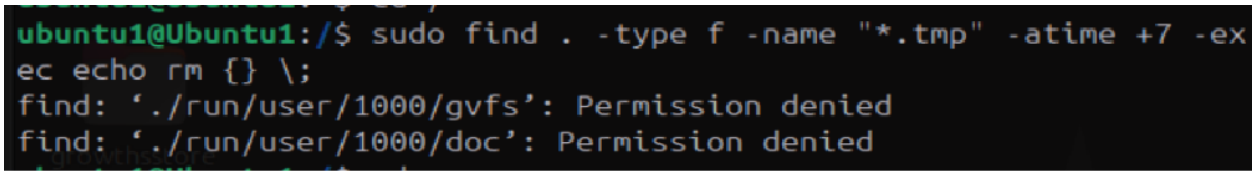


Showing how to preview dangerous operations before executing them.

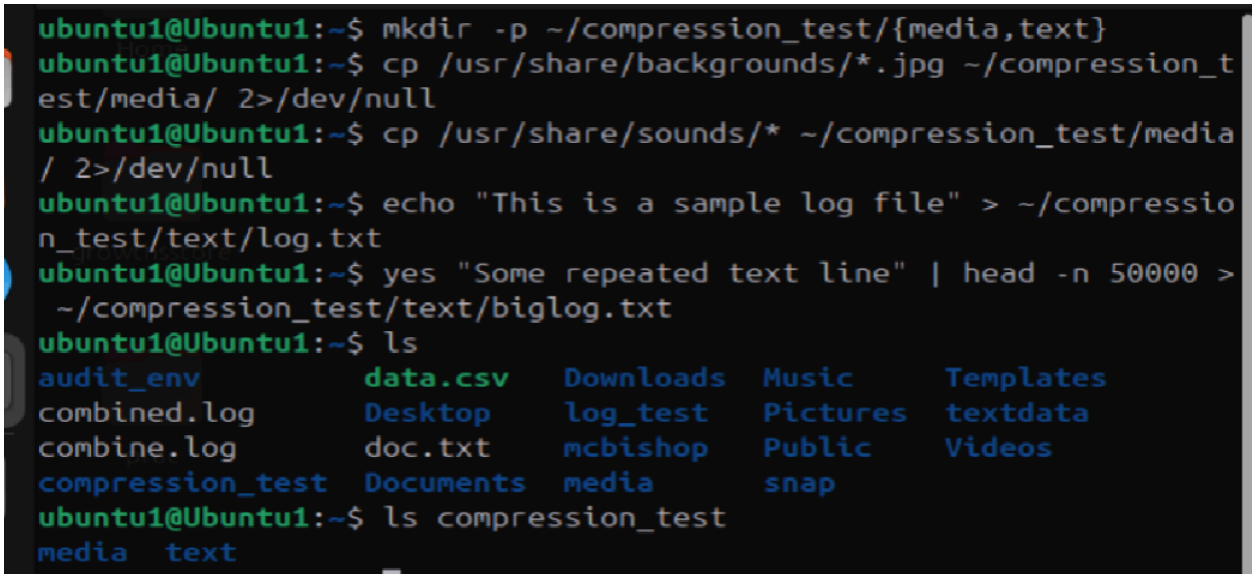
Always **preview with find first** before -exec.

Use -ok instead of -exec for destructive commands.

You can replace dangerous commands with echo first to verify:

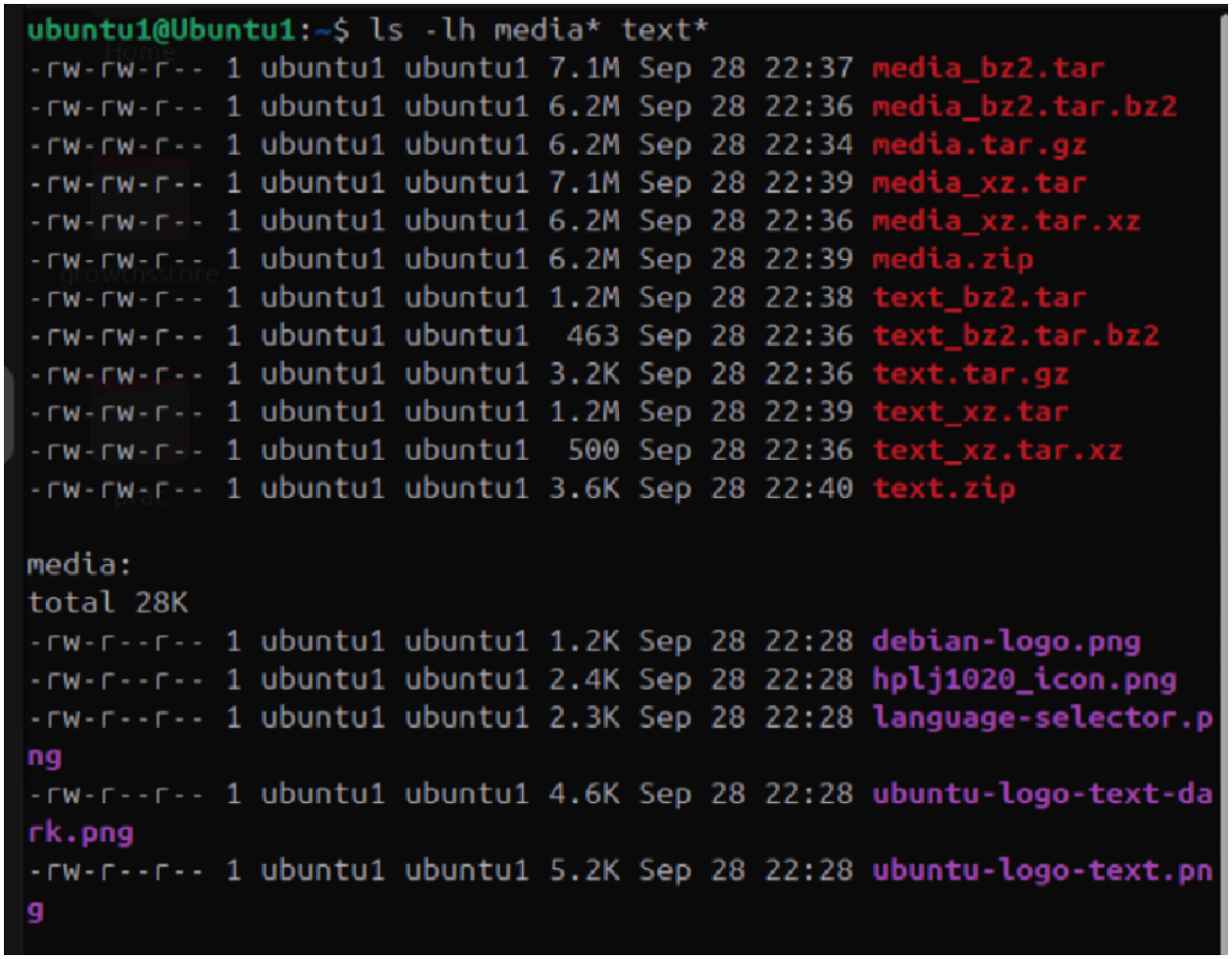


11)Create directories with different file types and sizes



Creating archives with different algorithms

Measuring sizes



Measuring speed (compression time)

```
ubuntu1@Ubuntu1:~$ time tar -czf test_text.tar.gz ~/compression_te
st/text
tar: Removing leading `/' from member names

real    0m0.070s
user    0m0.016s
sys     0m0.023s
ubuntu1@Ubuntu1:~$ time tar -cjf test_text.tar.bz2 ~/compression_t
est/text
tar: Removing leading `/' from member names

real    0m0.601s
user    0m0.133s
sys     0m0.203s
ubuntu1@Ubuntu1:~$ time tar -cJf test_text.tar.xz ~/compression_te
st/text
tar: Removing leading `/' from member names

real    0m0.086s
user    0m0.036s
sys     0m0.027s
ubuntu1@Ubuntu1:~$ time zip -r test_text.zip ~/compression_test/te
xt
updating: home/ubuntu1/compression_test/text/ (stored 0%)
updating: home/ubuntu1/compression_test/text/log.txt (stored 0%)
```

results

- Media (.jpg, .mp4, .zip): Already compressed all methods show little/no size reduction, only wasted CPU time. Best to just archive (tar without compression).
- Text files (logs, configs):

  gzip: Fast, good compression.

  bzip2: Better compression but slower.

  xz: Best compression ratio but slowest.

zip: Cross-platform, moderate compression.

Recommendation for automated backups

- Use tar + gzip (.tar.gz): good balance of speed + compression, widely supported.
- For long-term archival storage (rare restores, max space-saving) tar + xz.

For media files just tar (no compression) to save time/CPU.

12)Safely examining archive contents

```
ubuntu1@Ubuntu1:~$ tar -tzf logs.tar.gz
logs/
logs/log1.txt
logs/log2.txt
ubuntu1@Ubuntu1:~$ tar -tjf configs.tar.bz2
configs/
configs/db.conf
configs/app.conf
ubuntu1@Ubuntu1:~$ unzip -l media.zip
Archive:  media.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  2025-09-28 22:32   home/ubuntu1/compression_test/media/
  1408323  2025-09-28 22:32   home/ubuntu1/compression_test/media/
Province_of_the_south_of_france_by_orbitelambda.jpg
  4032506  2025-09-28 22:32   home/ubuntu1/compression_test/media/
Clouds_by_Tibor_Mokanszki.jpg
  1957897  2025-09-28 22:32   home/ubuntu1/compression_test/media/
Monument_valley_by_orbitelambda.jpg
        0  2025-09-28 23:15   home/ubuntu1/archives_demo/media/
       10  2025-09-28 23:15   home/ubuntu1/archives_demo/media/med
ia.txt
---------                     -------
  7398736                     6 files
```

Extracting only matching files

```
ubuntu1@Ubuntu1:~$ tar -xjf configs.tar.bz2 --wildcards '*.conf'
ubuntu1@Ubuntu1:~$ ls configs/*.conf
configs/app.conf  configs/db.conf
ubuntu1@Ubuntu1:~$ unzip media.zip '*/media.txt'
Archive:  media.zip
 extracting: home/ubuntu1/archives_demo/media/media.txt
ubuntu1@Ubuntu1:~$ cat extracted/*/media.txt
cat: 'extracted/*/media.txt': No such file or directory
ubuntu1@Ubuntu1:~$ cat archives_demo/media/media.txt
mediafile
```

Updating existing archives without full recreation

```
ubuntu1@Ubuntu1:~$ tar -rf logs.tar logs/log3.txt    # append
gzip logs.tar
tar: logs/log3.txt: Cannot stat: No such file or directory
tar: Exiting with failure status due to previous errors
gzip: logs.tar.gz already exists; do you wish to overwrite (y or n
)? y
ubuntu1@Ubuntu1:~$ tar -tzf logs.tar.gz
```

Handling corrupted archives

```
ubuntu1@Ubuntu1:~$ cp logs.tar.gz broken.tar.gz
ubuntu1@Ubuntu1:~$ truncate -s -10 broken.tar.gz   # chop off last
 bytes
ubuntu1@Ubuntu1:~$ tar -tzf broken.tar.gz

gzip: stdin: unexpected end of file
tar: Child returned status 1
tar: Error is not recoverable: exiting now
ubuntu1@Ubuntu1:~$ gzip -tv broken.tar.gz
broken.tar.gz:
gzip: broken.tar.gz: unexpected end of file
```

Merging contents into a single new archive

```
ubuntu1@Ubuntu1:~$ ls -R ~/merged
/home/ubuntu1/merged:
configs  home

/home/ubuntu1/merged/configs:
app.conf  db.conf

/home/ubuntu1/merged/home:
ubuntu1

/home/ubuntu1/merged/home/ubuntu1:
archives_demo  compression_test

/home/ubuntu1/merged/home/ubuntu1/archives_demo:
media

/home/ubuntu1/merged/home/ubuntu1/archives_demo/media:
media.txt

/home/ubuntu1/merged/home/ubuntu1/compression_test:
```

13)Your approach must handle daily incremental backups, weekly full backups, monthly archives for long-term storage

```
ubuntu1@Ubuntu1:~$ mkdir -p ~/prod_data
ubuntu1@Ubuntu1:~$ echo "file1" > ~/prod_data/app.log
ubuntu1@Ubuntu1:~$ echo "file2" > ~/prod_data/config.cfg
ubuntu1@Ubuntu1:~$ ls -l ~/prod_data
total 8
-rw-rw-r-- 1 ubuntu1 ubuntu1 6 Sep 29 14:12 app.log
-rw-rw-r-- 1 ubuntu1 ubuntu1 6 Sep 29 14:12 config.cfg
```

```
ubuntu1@Ubuntu1:~$ mkdir -p ~/backups/{daily,weekly,monthly}
ubuntu1@Ubuntu1:~$ ls -R ~/backups
/home/ubuntu1/backups:
daily  monthly  weekly

/home/ubuntu1/backups/daily:

/home/ubuntu1/backups/monthly:

/home/ubuntu1/backups/weekly:
```

```
ubuntu1@Ubuntu1:~$ mkdir -p ~/data
ubuntu1@Ubuntu1:~$ echo "file1" > ~/data/file1.txt
ubuntu1@Ubuntu1:~$ echo "file2" > ~/data/file2.txt
ubuntu1@Ubuntu1:~$ ls -l ~/data
total 8
-rw-rw-r-- 1 ubuntu1 ubuntu1 6 Sep 29 14:32 file1.txt
-rw-rw-r-- 1 ubuntu1 ubuntu1 6 Sep 29 14:33 file2.txt
```

Daily incremental backup

```
ubuntu1@Ubuntu1:~$ # 1. Make sure full structure exists
mkdir -p ~/backups/daily

# 2. Create the snapshot file
touch ~/backups/daily.snar

# 3. Run incremental backup again
tar --listed-incremental=~/backups/daily.snar -czpf ~/backups/
daily/backup_$(date +%F).tar.gz ~/data

# 4. Verify contents
tar -tzf ~/backups/daily/backup_$(date +%F).tar.gz
tar: ~/backups/daily.snar: Cannot open: No such file or direct
ory
tar: Removing leading `/' from member names
tar: Exiting with failure status due to previous errors
home/ubuntu1/data/
home/ubuntu1/data/file1.txt
home/ubuntu1/data/file2.txt
```

Weekly full backup

```
ubuntu1@Ubuntu1:~$ tar --listed-incremental=/dev/null -czpf ~/
backups/weekly/full_$(date +%F).tar.gz ~/data
tar: Removing leading `/' from member names
tar: /dev/null: Cannot truncate: Invalid argument
tar: Exiting with failure status due to previous errors
ubuntu1@Ubuntu1:~$ tar -tzf ~/backups/weekly/full_$(date +%F).
tar.gz
home/ubuntu1/data/
home/ubuntu1/data/file1.txt
home/ubuntu1/data/file2.txt
```

MonthlyArchive

```
ubuntu1@Ubuntu1:~$ tar -czpf ~/backups/monthly/archive_$(date
+%Y-%m).tar.gz ~/data
tar: Removing leading `/' from member names
ubuntu1@Ubuntu1:~$ tar -tzf ~/backups/monthly/archive_$(date +
%Y-%m).tar.gz
home/ubuntu1/data/
home/ubuntu1/data/file2.txt
home/ubuntu1/data/file1.txt
```

Automatic cleanup (keep last 7 daily, 4 weekly, 12 monthly)

```
ubuntu1@Ubuntu1:~$ cd home
ubuntu1@Ubuntu1:~/home$ ls
ubuntu1
ubuntu1@Ubuntu1:~/home$ find ~/backups/daily -type f -mtime +7
 -exec echo rm {} \;
ubuntu1@Ubuntu1:~/home$ find ~/backups/weekly -type f -mtime +
28 -exec echo rm {} \;
ubuntu1@Ubuntu1:~/home$ find ~/backups/monthly -type f -mtime
+365 -exec echo rm {} \;
ubuntu1@Ubuntu1:~/home$ find ~/backups/daily -type f -mtime +7
ubuntu1@Ubuntu1:~/home$ find ~/backups/weekly -type f -mtime +
28find ~/backups/monthly -type f -mtime +365
find: invalid argument `+28find' to `-mtime'
ubuntu1@Ubuntu1:~/home$ find ~/backups/monthly -type f -mtime
+365
```

Verifying backup integrity

```
ubuntu1@Ubuntu1:~$ tar -tzf ~/backups/daily/backup_$(date +%F)
.tar.gz > /dev/null && echo "OK"
OK
ubuntu1@Ubuntu1:~$ tar -tzf ~/backups/daily/backup_$(date +%F)
.tar.gz > /dev/null && echo "OK"
OK
```

14) Compare your user's groups with another user account's groups (create a test scenario)

```
ubuntu1@Ubuntu1:~$ whoami
ubuntu1
ubuntu1@Ubuntu1:~$ id
uid=1000(ubuntu1) gid=1000(ubuntu1) groups=1000(ubuntu1),4(a
dm),24(cdrom),27(sudo),30(dip),46(plugdev),100(users),114(lp
admin)
ubuntu1@Ubuntu1:~$ groups
ubuntu1 adm cdrom sudo dip plugdev users lpadmin
ubuntu1@Ubuntu1:~$
```

Creating a test user to compare groups

```
ubuntu1@Ubuntu1:~$ sudo useradd -m testuser
[sudo] password for ubuntu1:
ubuntu1@Ubuntu1:~$ id testuser
uid=1002(testuser) gid=1002(testuser) groups=1002(testuser)
ubuntu1@Ubuntu1:~$ groups testuser
testuser : testuser
```

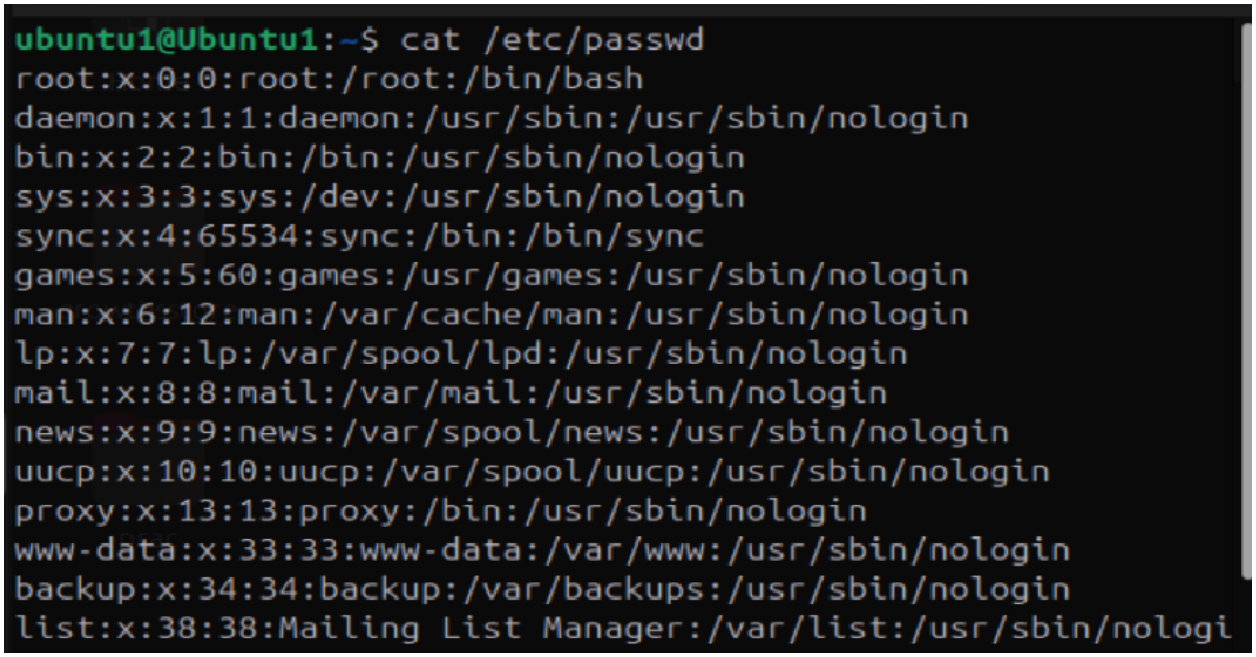
```
ubuntu1@Ubuntu1:~$ groups
ubuntu1 adm cdrom sudo dip plugdev users lpadmin
ubuntu1@Ubuntu1:~$ groups testuser
testuser : testuser
```

Observation: system accounts usually have limited groups, regular users often in sudo, users

Examining /etc/passwd entries

System users: UID < 1000, login shell usually /usr/sbin/nologin or /bin/false, no home directory for normal login.

Regular users: UID ≥ 1000, valid shell (e.g., /bin/bash), home directory /home/username



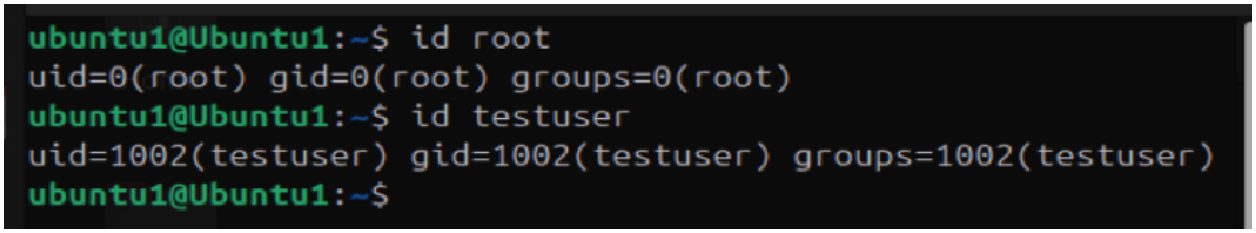
- Comparing groups with a system user

  Risk: If a regular user is added to groups like root or adm, they could:

  Read/write sensitive files

  Change system configurations

  Escalate privileges



15)Checking current effective vs configured groups

```
ubuntu1@Ubuntu1:~$ id
uid=1000(ubuntu1) gid=1000(ubuntu1) groups=1000(ubuntu1),4(a
dm),24(cdrom),27(sudo),30(dip),46(plugdev),100(users),114(lp
admin)
ubuntu1@Ubuntu1:~$ groups
ubuntu1 adm cdrom sudo dip plugdev users lpadmin
ubuntu1@Ubuntu1:~$
```

Demonstrating re-login requirement

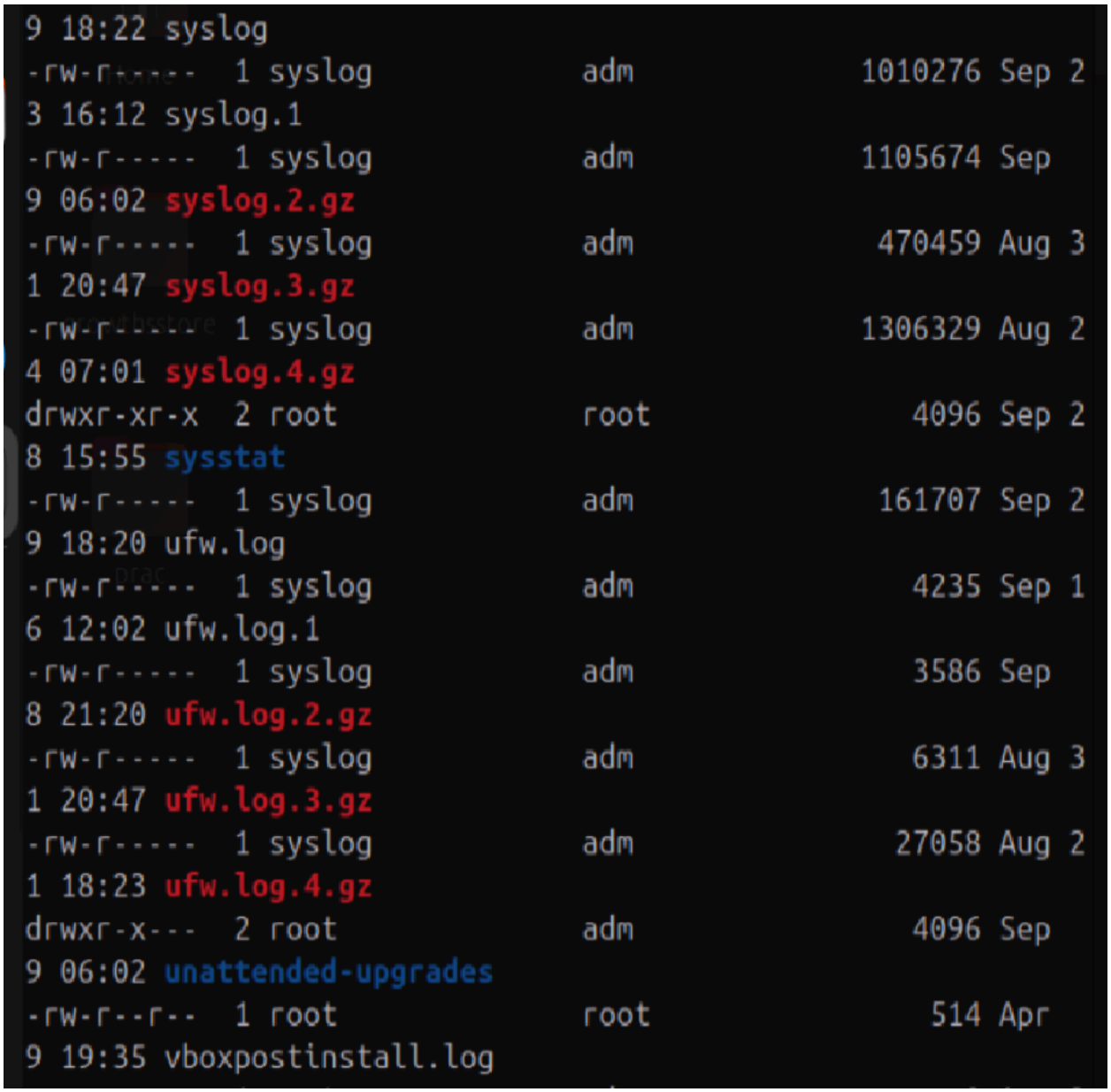
```
ubuntu1@Ubuntu1:~$ sudo useradd -m colleague
ubuntu1@Ubuntu1:~$ sudo passwd ubuntu
passwd: user 'ubuntu' does not exist
ubuntu1@Ubuntu1:~$ sudo passwd colleague
New password:
BAD PASSWORD: The password is shorter than 8 characters
Retype new password:
passwd: password updated successfully
ubuntu1@Ubuntu1:~$ sudo usermod -aG adm colleague
ubuntu1@Ubuntu1:~$ id colleague
uid=1003(colleague) gid=1003(colleague) groups=1003(colleagu
e),4(adm)
ubuntu1@Ubuntu1:~$ groups colleague
colleague : colleague adm
ubuntu1@Ubuntu1:~$ groups
ubuntu1 adm cdrom sudo dip plugdev users lpadmin
```

After relog-in

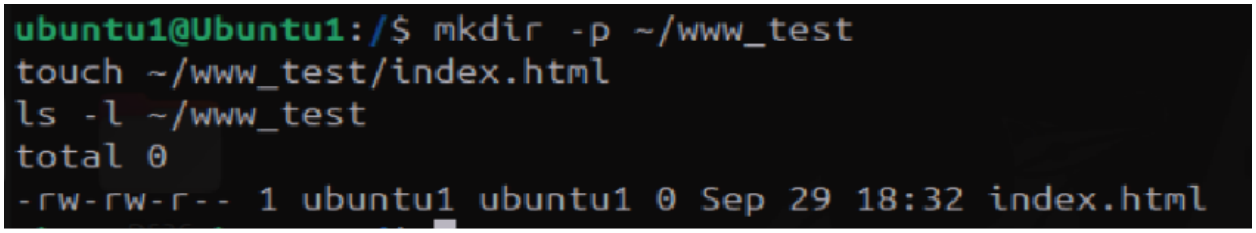
```
ubuntu1@Ubuntu1:~$ id colleague
uid=1003(colleague) gid=1003(colleague) groups=1003(colleague
),4(adm)
ubuntu1@Ubuntu1:~$ groups colleague
colleague : colleague adm
ubuntu1@Ubuntu1:~$
```

Identifying groups for common resources(Also checking group ownership and web files

Group ownership


9 18:22 syslog
-rw-r----- 1 syslog adm 1010276 Sep 2
3 16:12 syslog.1
-rw-r----- 1 syslog adm 1105674 Sep
9 06:02 syslog.2.gz
-rw-r----- 1 syslog adm 470459 Aug 3
1 20:47 syslog.3.gz
-rw-r----- 1 syslog adm 1306329 Aug 2
4 07:01 syslog.4.gz
drwxr-xr-x 2 root root 4096 Sep 2
8 15:55 sysstat
-rw-r----- 1 syslog adm 161707 Sep 2
9 18:20 ufw.log
-rw-r----- 1 syslog adm 4235 Sep 1
6 12:02 ufw.log.1
-rw-r----- 1 syslog adm 3586 Sep
8 21:20 ufw.log.2.gz
-rw-r----- 1 syslog adm 6311 Aug 3
1 20:47 ufw.log.3.gz
-rw-r----- 1 syslog adm 27058 Aug 2
1 18:23 ufw.log.4.gz
drwxr-x--- 2 root adm 4096 Sep
9 06:02 unattended-upgrades
-rw-r--r-- 1 root root 514 Apr
9 19:35 vboxpostinstall.log

Web files

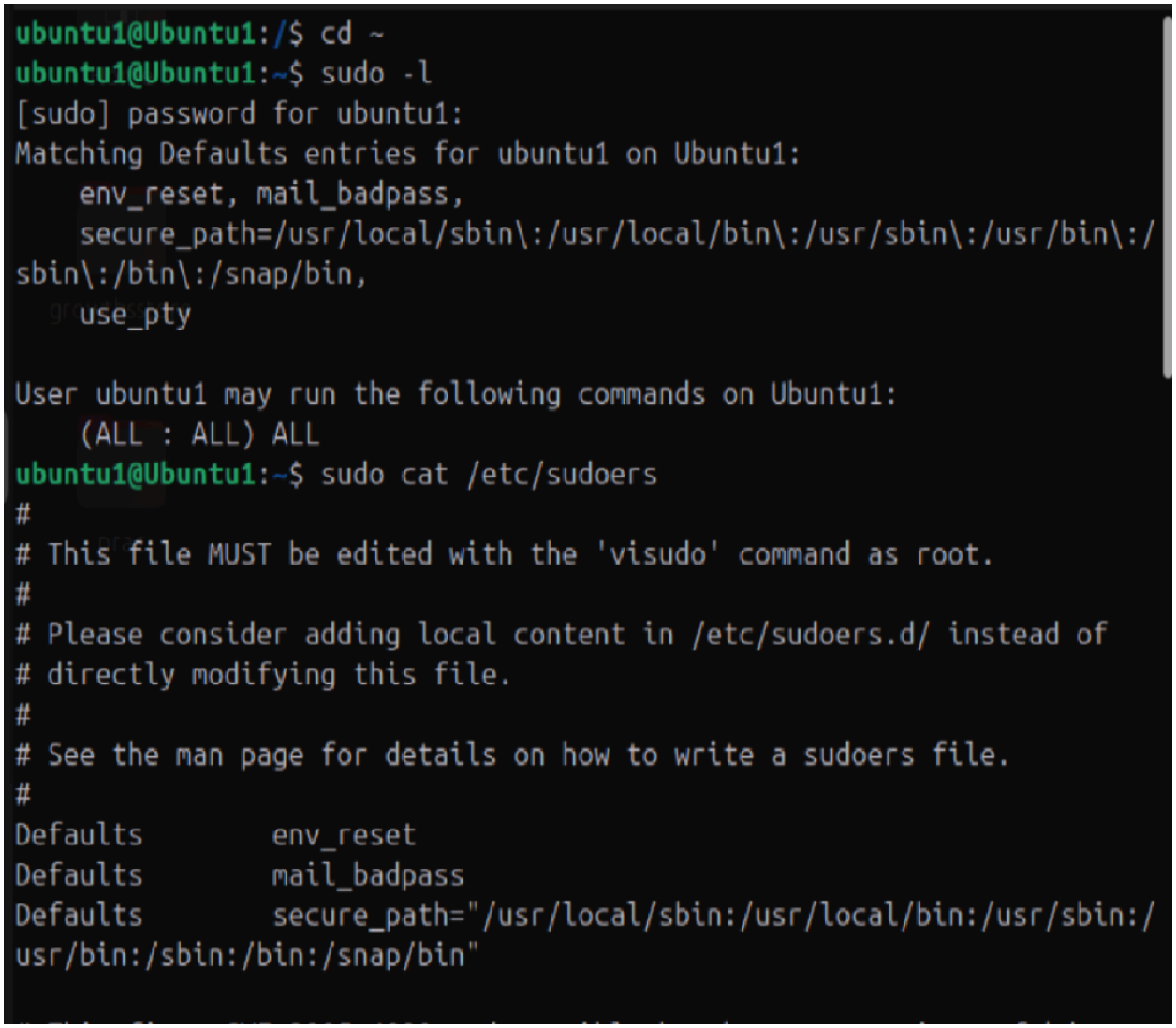


Assign users **only to the groups they need** for their job.

Avoid giving access to root, adm, or sudo unless necessary.

Helps **limit potential damage** from mistakes or compromised accounts.

16)Documenting my sudo permissions & restrictions



Showing what my user is allowed to do and inspect sudoers safely.

```
# While you shouldn't normally run git as root, you need to with etcke
per
#Defaults:%sudo env_keep += "GIT_AUTHOR_* GIT_COMMITTER_*"

# Per-user preferences; root won't have sensible values for them.
#Defaults:%sudo env_keep += "EMAIL DEBEMAIL DEBFULLNAME"

# "sudo scp" or "sudo rsync" should be able to use your SSH agent.
#Defaults:%sudo env_keep += "SSH_AGENT_PID SSH_AUTH_SOCK"

# Ditto for GPG agent
#Defaults:%sudo env_keep += "GPG_AGENT_INFO"

# Host alias specification

# User alias specification

# Cmnd alias specification

# User privilege specification
root    ALL=(ALL:ALL) ALL

# Members of the admin group may gain root privileges
%admin ALL=(ALL) ALL
```

Demonstrating difference: sudo -i vs sudo su vs su

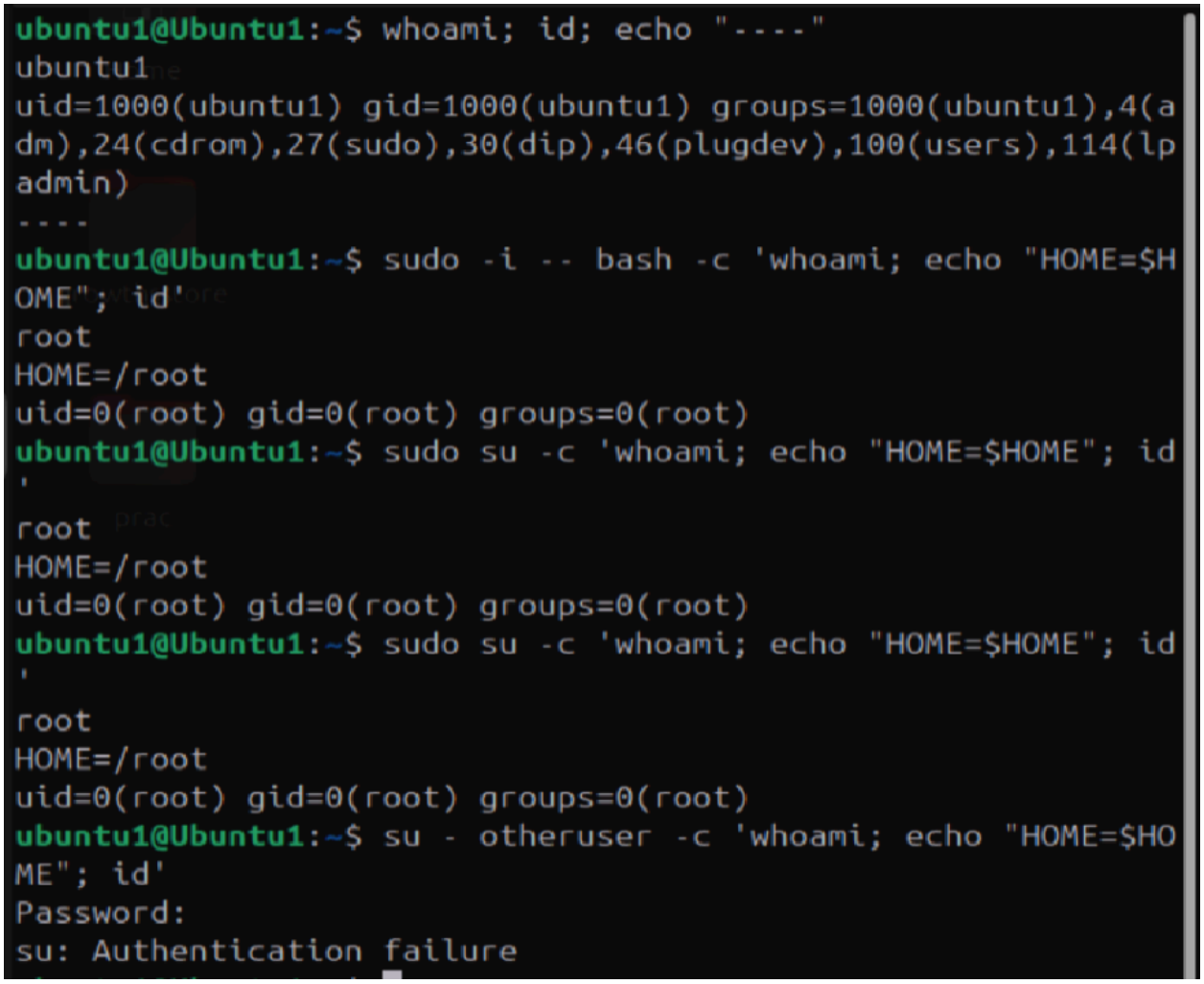
Each command below prints who you are, your HOME, and effective groups so you can inspect differences.

sudo -i : start a login shell as root (reads root's shell startup files)

sudo su : run su as root; typically gives a root shell (behavior depends on su flags)

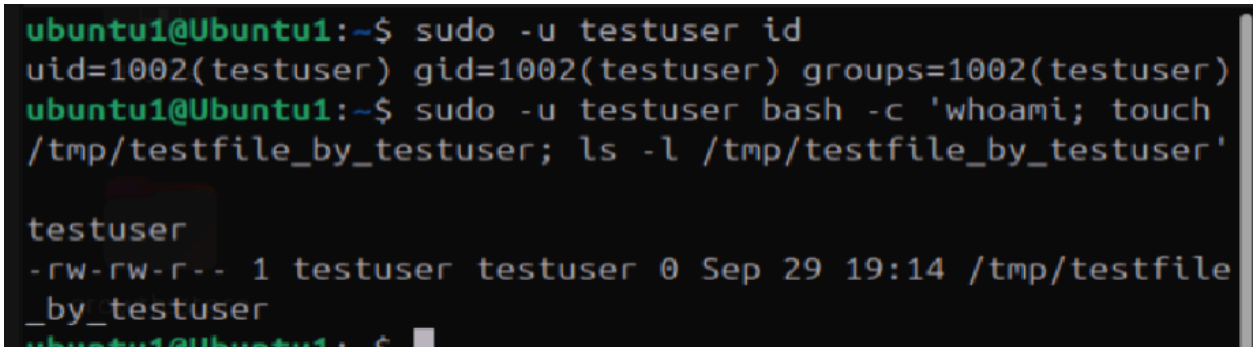
su : switch user (requires target user's password unless run as root)

Example: run a command as another user "otheruser" (may prompt for password)



Run commands as specific users (not root)

Examples to execute a command as testuser (created recently)
Run a single command as testuser (using sudo)



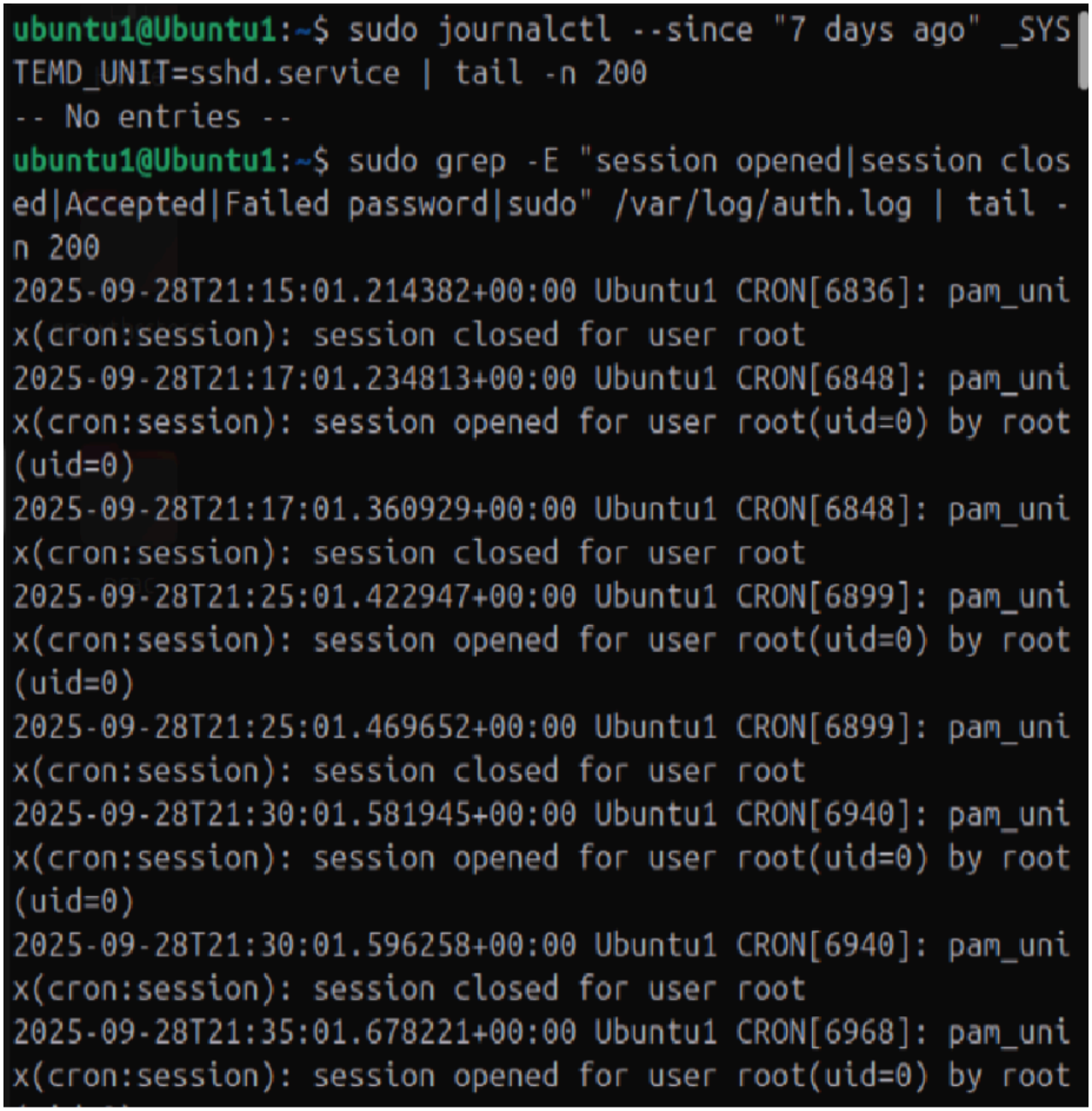
Analyzing login / sudo patterns using logs

Commands to extract recent relevant entries

For systemd systems: SSH/session journal (last 7 days)

Show only sudo usage entries

List recent successful interactive logins



```
ubuntu1@Ubuntu1:~$ sudo last -a | head -n 20
ubuntu1  tty2         Mon Sep 29 18:19   still logged in
tty2
ubuntu1  seat0        Mon Sep 29 18:19   still logged in
login screen
reboot   system boot  Mon Sep 29 18:16   still running
6.11.0-24-generic
colleagu tty3         Mon Sep 29 16:56 - crash  (01:19)
tty3
colleagu seat0        Mon Sep 29 16:56 - crash  (01:19)
login screen
ubuntu1  tty2         Mon Sep 29 16:54 - crash  (01:21)
tty2
ubuntu1  seat0        Mon Sep 29 16:54 - 16:56  (00:02)
login screen
reboot   system boot  Mon Sep 29 16:54   still running
6.11.0-24-generic
ubuntu1  tty2         Mon Sep 29 15:56 - crash  (00:58)
tty2
ubuntu1  seat0        Mon Sep 29 15:56 - crash  (00:58)
login screen
ubuntu1  tty2         Mon Sep 29 14:09 - 15:56  (01:46)
tty2
ubuntu1  seat0        Mon Sep 29 14:09 - 15:56  (01:46)
login screen
reboot   system boot  Mon Sep 29 14:08   still running
```