You completed this test on *2026/02/11, 18:53*
Your score is 85.00%

CORRECT

1. You need a symbolic link named /usr/local/bin/tool that always points to ../opt/tool/bin/tool relative to the link's location, and you must overwrite any existing destination (even if it is a symlink to a directory) without following it. Which command best satisfies this on GNU ln?

ln -s ../opt/tool/bin/tool /usr/local/bin/tool

✓ *ln -sfnT ../opt/tool/bin/tool /usr/local/bin/tool*

ln -sf ../opt/tool/bin/tool /usr/local/bin/tool

ln -sF ../opt/tool/bin/tool /usr/local/bin/tool

CORRECT

2. In a directory /a, you want to create symlinks in /b for every file in /a, but each symlink must be relative (so moving /b preserves link validity as long as relative layout stays). Which option enables relative targets?

ln -s -t /b /a/*

✓ *ln -s --relative -t /b /a/**

ln --logical -s -t /b /a/*

ln -s --dereference -t /b /a/*

INCORRECT

3. You run: ln -s existing_dir linkname where linkname already exists and is a directory. You want the link itself to replace the directory (not to create a link inside it). Which combination is required on GNU ln?

ln -sTf existing_dir linkname

✗ *ln -sT existing_dir linkname*

ln -sf existing_dir linkname

ln -sF existing_dir linkname

CORRECT

4. Which statement about hard links created with ln is correct on a typical Linux ext4 filesystem?

Hard links can span filesystems if you use ln -L

Hard links can be made to directories by root using ln -d

✓ *Hard links refer to the same inode and share link count; removing one name does not remove the data until the last link is removed*

Hard links always record the original path so they break if the target is moved

CORRECT

5. You must change ownership of a symlink itself (not the referenced file) to user:group, and do so recursively under a directory tree without dereferencing symlinks encountered. Which is the most correct GNU chown invocation?

chown -R user:group /path

✓ *chown -Rh user:group /path*

chown -R --dereference user:group /path

chown -RP user:group /path

CORRECT

6. You want to change owner from alice to bob only for files currently owned by alice (leave others untouched), including group unchanged. Which command achieves that?

chown bob --from=alice /path

✓ *chown --from=alice bob /path*

chown --from=alice: bob /path

chown bob: --from=alice /path

CORRECT

7. You need to copy ownership (user and group) from /template/file to /data/target without specifying explicit names. Which option is designed for this?

chown --mirror=/template/file /data/target

✓ *chown --reference=/template/file /data/target*

chown --inherit=/template/file /data/target

chown --clone=/template/file /data/target

CORRECT

8. On GNU chown, what is the effect of specifying owner and group as ':developers' (leading colon) like: chown :developers file ?

Sets both owner and group to developers

✓ *Sets only the group to developers; owner remains unchanged*

Sets only the owner to developers; group remains unchanged

Fails unless the current owner is root

INCORRECT

9. A file currently has mode 2751 (setgid + rwxr-x--x). You run: chmod u=rwX,g=u,o= file. What is the resulting mode?

2670

✗ *2760*

2750

2751

CORRECT

10. You need to recursively add execute permission only to directories (not to regular files) under /srv/app, while leaving existing permissions otherwise untouched. Which is best?

✓ *chmod -R a+X /srv/app*

chmod -R a+x /srv/app

chmod -R +X /srv/app

chmod -R u+X,g+X,o+X /srv/app

INCORRECT

11. You must make /bin/tool setuid-root while removing write permission for group and others, without changing any other bits that are currently set for user. Which symbolic mode fits?

chmod u+s,go-w /bin/tool

chmod 4755 /bin/tool

chmod u=rsx,go=rx /bin/tool

✗ *chmod u+s,go=rx /bin/tool*

CORRECT

12. Which statement about chmod's numeric modes is true?

The leading digit 2 sets the setuid bit

The leading digit 4 sets the sticky bit

✓ *The leading digit 1 sets the sticky bit*

The leading digit 8 enables ACL inheritance

CORRECT

13. You want to delete all *.tmp files under /var/log EXCEPT those under /var/log/keep. Which find expression is correct and avoids descending into keep?

✓ *find /var/log -name keep -prune -o -name '*.tmp' -delete*

find /var/log -path /var/log/keep -prune -and -name '*.tmp' -delete

find /var/log -name '*.tmp' -delete -path /var/log/keep -prune

find /var/log -name '*.tmp' -delete -not -path '/var/log/keep/*'

CORRECT

14. Given: find . -name '*.log' -print -o -name '*.bak' -print. Why might this output .bak files even when a .log match is found?

Because -print is implicit and forces evaluation of all terms

Because -o has lower precedence than -print, so both branches may run

✓ *Because -print returns true, and -o short-circuits only when the left side is true for the same file*

Because -name caches results and reuses them

CORRECT

15. You need to search for files modified in the last 36 hours. Which predicate is most accurate?

find . -mtime -1.5

find . -mtime -2

✓ *find . -mmin -2160*

find . -ctime -2160

INCORRECT

16. You want find to execute a command with as many path arguments as possible per process for efficiency, while still safely handling spaces/newlines in filenames. Which construct is appropriate?

-exec cmd {} ;

-exec cmd {} +

✗ *| xargs cmd*

-ok cmd {} +

CORRECT

17. You must search a binary-ish log that may contain NUL bytes and you want grep to treat the entire file as text and allow matches across NUL-separated records. Which option combination is designed for NUL data and prints only the match?

grep -a -o pattern file

✓ *grep -z -o pattern file*

grep -U -o pattern file

grep -b -o pattern file

CORRECT

18. You need to match the string 'foo.bar' literally (dot should not be regex wildcard) in multiple files, but you also need line numbers. Which is correct?

grep -n 'foo.bar' *.txt

✓ *grep -n -F 'foo.bar' *.txt*

grep -n -E 'foo\.bar' *.txt

grep -n -P 'foo\\.bar' *.txt

CORRECT

19. In GNU grep, what does option -m 1 do?

✓ *Stop reading each file after 1 match is found (per file)*

Stop after 1 match total across all files

Print only the first matching group

Match only 1 character

CORRECT

20. You want to display 2 lines before and 3 lines after each match, while also showing a group separator only between distinct match groups. Which is best?

grep -B2 -A3 --no-group-separator pattern file

grep -C2,3 pattern file

✓ *grep -B2 -A3 --group-separator='--' pattern file*

grep -B2 -A3 -m1 pattern file

INCORRECT

21. You want the first 1024 bytes of a file, regardless of line boundaries. Which command is correct?

head -n 1024 file

head -c 1024 file

✗ *head -b 1024 file*

head --bytes=1024 --lines=0 file

CORRECT

22. What does GNU head -n -5 file output?

The last 5 lines

✓ *All but the last 5 lines*

The first 5 lines

Nothing; negative counts are invalid

INCORRECT

23. You are combining head with a pipeline and need to avoid noisy 'Broken pipe' diagnostics in scripts when the downstream command exits early. Which head option is relevant?

✗ *--quiet*

--silent

--verbose

--zero-terminated

CORRECT

24. Given a file with NUL-separated records, you want the first 10 records (not "lines"). Which option makes head treat NUL as the record delimiter?

✓ *-z*

-0

--nul

--record

CORRECT

25. You need to follow a log file that may be rotated (renamed and recreated). Which tail option is specifically intended to follow by name across rotations?

tail -f

✓ *tail -F*

tail --follow=descriptor

tail -r

CORRECT

26. You want tail to stop following when a specific PID exits (GNU tail). Which option is designed for this?

✓ *--pid=PID*

--until=PID

--exit-pid PID

--stop=PID

CORRECT

27. What does GNU tail -n +10 file do?

Print the last 10 lines

✓ *Start printing from line 10 to the end*

Print exactly 10 lines

Print all but the last 10 lines

CORRECT

28. You want the last 512 bytes of a file. Which is correct?

tail -n 512 file

✓ *tail -c 512 file*

tail -b 512 file

tail --lines=0 --bytes=-512 file

CORRECT

29. You want a custom output listing PID, PPID, elapsed time, and the full command line, sorted by descending elapsed time. Which ps invocation is most appropriate (procps)?

✓ *ps -e -o pid,ppid,etime,args --sort=-etime*

ps aux --sort=-etime --format pid,ppid,etime,args

ps -ef --sort etime -o pid ppid etime cmd

ps -eo pid,ppid,etimes,cmd --sort=etime

CORRECT

30. In procps ps, what is the key difference between 'args' and 'cmd' output specifiers?

✓ *args shows the full command line; cmd shows the executable name in brackets for kernel threads*

cmd shows the full command line; args shows only the basename

They are identical aliases

args includes environment variables; cmd does not

CORRECT

31. You need to show threads (LWPs) for a given process PID 1234. Which option is most direct?

ps -p 1234 -o pid,tid,cmd

✓ *ps -T -p 1234*

ps --threads 1234

ps -L 1234

CORRECT

32. Which statement about ps option styles is correct on Linux?

BSD-style options require a leading '-'

UNIX options cannot be mixed with BSD options

✓ *GNU ps supports three styles (Unix, BSD, and GNU long options), and the meaning of some options can differ between styles*

Only System V style is supported in procps

CORRECT

33. You want to test whether process 4242 exists and you have permission to signal it, without sending any actual signal. Which command is correct?

✓ *kill -0 4242*

kill -NULL 4242

kill -test 4242

kill --check 4242

CORRECT

34. Which command sends SIGTERM to process group 1234 (all processes in that group) using kill syntax?

kill -TERM 1234

kill -- -1234

✓ *kill -TERM -1234*

kill -g 1234

CORRECT

35. What does 'kill -l' typically do?

Lists running processes

✓ *Lists signal names (or converts between numbers and names)*

Sends SIGKILL to the last process

Logs signals delivered to syslog

CORRECT

36. A process ignores SIGTERM and continues running. Which signal is conventionally used next as a last resort because it cannot be caught or ignored?

SIGHUP

SIGINT

✓ *SIGKILL*

SIGSTOP

CORRECT

37. In many shells, echo is a builtin and its behavior can vary. Which practice is most reliable for printing arbitrary strings that may start with '-' or contain backslash escapes, without interpreting them?

echo -E -- "$string"

echo -- "$string"

✓ *printf '%s\n' "$string"*

echo -n "$string"

CORRECT

**38. Given: echo -e 'A
B' What is printed by a typical Bash builtin echo?**

A\nB (literal backslash-n)

✓ *A then a newline then B*

A then the letter n then B

Nothing (invalid escape)

CORRECT

39. Which option to echo suppresses the trailing newline in common implementations?

-s

✓ *-n*

-N

--no-newline

CORRECT

40. You run: echo $'X\x41Y' Assuming Bash $'...' ANSI-C quoting, what is the output?

X\x41Y

✓ *XAY*

X\A Y

XA\nY