Perfect 👍 Let's turn this into a full exercise + solution set. I'll list each exercise with one or more solutions (including alternatives where possible).

Because it's a big set, I'll cover Easy first. If this format works for you, I'll continue with Medium and Hard in the same way.

🐧 Linux & Shell Scripting — Exercises with Solutions

🔰 Easy Level

File & Directory Management

1. Create a directory called **`practice`** and navigate into it.

```
mkdir practice
cd practice
```

2. Create an empty file named **`notes.txt`** inside **`practice`**.

```
touch notes.txt
# Alternative
> notes.txt
```

3. Copy **`notes.txt`** into a new file called **`backup.txt`**.

```
cp notes.txt backup.txt
```

4. Move **`backup.txt`** into a new folder called **`archive`**.

```
mkdir archive
mv backup.txt archive/
```

5. Delete **`backup.txt`** and remove the **`archive`** directory.

```
rm archive/backup.txt
rmdir archive
# Alternative (force remove non-empty folder)
rm -r archive
```

Viewing & Searching

6. Display the first 10 lines of **`/etc/passwd`**.

```
head /etc/passwd
# Alternative
sed -n '1,10p' /etc/passwd
```

7. Display the last 15 lines of **`/var/log/syslog`**.

```
tail -n 15 /var/log/syslog
# Alternative
sed -n -e :a -e '$q;N;16,$D;ba' /var/log/syslog
```

8. Count the number of words in **notes.txt**.

```
wc -w notes.txt
```

9. Search for the word "root" in **/etc/passwd**.

```
grep root /etc/passwd
# Case insensitive
grep -i root /etc/passwd
```

10. Find all **.txt** files inside your home directory.

```
find ~ -type f -name "*.txt"
# Alternative
ls ~/*.txt 2>/dev/null
```

Permissions & Processes

11. Change permissions of **notes.txt** to **rw-r--r--**.

```
chmod 644 notes.txt
```

12. Make yourself the owner of **notes.txt**.

```
sudo chown $USER notes.txt
```

13. Run a background process using **sleep 100 &**.

```
sleep 100 &
```

14. List all running processes and kill the sleep process.

```
ps aux | grep sleep
kill <PID>
# Alternative (kill all sleep)
killall sleep
```

Basic Scripting

15. Write a script that prints "Hello, Linux!" to the terminal. File: *hello.sh*

```
#!/bin/bash
echo "Hello, Linux!"
```

Run:

```
bash hello.sh
```

16. Write a script that takes your name as input and greets you. File: *greet.sh*

```
#!/bin/bash
echo "Enter your name:"
read name
echo "Hello, $name!"
```

17. Write a script that displays today's date and time. File: *date.sh*

```
#!/bin/bash
echo "Today is: $(date)"
```

◈ Medium Level

File Handling & Text Processing

18. Rename all **.txt** files in a folder to **.bak**.

```
for file in *.txt; do
    mv "$file" "${file%.txt}.bak"
done
# Alternative using rename
rename 's/\.txt$/.bak/' *.txt
```

19. Display the top 5 largest files in your home directory.

```
du -ah ~ | sort -rh | head -n 5
# Alternative (only files, not dirs)
find ~ -type f -exec du -h {} + | sort -rh | head -n 5
```

20. Count the number of **.sh** scripts in the current directory.

```
ls -1 *.sh 2>/dev/null | wc -l
# Alternative
find . -maxdepth 1 -type f -name "*.sh" | wc -l
```

21. Extract all usernames from **/etc/passwd** and save to **users.txt**.

```
cut -d: -f1 /etc/passwd > users.txt
# Alternative
awk -F: '{print $1}' /etc/passwd > users.txt
```

22. Check if a file exists and print a message.

```
#!/bin/bash
echo "Enter filename:"
read file
if [ -e "$file" ]; then
    echo "File exists"
else
```

```
    echo "File does not exist"
fi
```

Loops & Conditions

23. Print numbers 1 to 10 using a `for` loop.

```
for i in {1..10}; do
    echo $i
done
# Alternative
seq 10
```

24. Sum numbers from 1 to 100 using a `while` loop.

```
sum=0
i=1
while [ $i -le 100 ]; do
    sum=$((sum + i))
    i=$((i + 1))
done
echo "Sum = $sum"
```

25. Check if a given file is empty or not.

```
#!/bin/bash
read -p "Enter filename: " file
if [ ! -s "$file" ]; then
    echo "File is empty"
else
    echo "File is not empty"
fi
```

26. Print whether a number is even or odd.

```
#!/bin/bash
read -p "Enter number: " n
if (( n % 2 == 0 )); then
    echo "Even"
else
    echo "Odd"
fi
```

System & User Management

27. Display the top 5 CPU-consuming processes.

```
ps -eo pid,comm,%cpu --sort=-%cpu | head -n 6
```

28. Display the top 5 memory-consuming processes.

```
ps -eo pid,comm,%mem --sort=-%mem | head -n 6
```

29. Create a new user with a given username.

```
sudo adduser newusername
# Alternative
sudo useradd -m newusername && sudo passwd newusername
```

30. Monitor free disk space and warn if usage exceeds 80%.

```
#!/bin/bash
usage=$(df / | awk 'NR==2 {print $5}' | sed 's/%//')
if [ $usage -gt 80 ]; then
    echo "Warning: Disk usage above 80% ($usage%)"
else
    echo "Disk usage is $usage%"
fi
```

◈ Hard Level

Automation & File Management

31. Backup a directory into a `.tar.gz` with today's date.

```
#!/bin/bash
tar -czf backup_$(date +%F).tar.gz /path/to/directory
```

32. Monitor a directory and log whenever a new file is added.

```
#!/bin/bash
inotifywait -m /path/to/dir -e create |
while read path action file; do
    echo "$(date): $file was created in $path" >> dir_monitor.log
done
```

33. Compress all log files older than 7 days.

```
find /var/log -type f -name "*.log" -mtime +7 -exec gzip {} \;
```

34. Find and delete files larger than 100MB.

```
find /path/to/dir -type f -size +100M -delete
```

Text & Data Processing

35. Find top 10 IPs from Apache/Nginx log.

```
awk '{print $1}' /var/log/apache2/access.log | sort | uniq -c | sort -nr | head -n 10
```

36. Extract email addresses from a text file.

```
grep -E -o "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-z]{2,}" file.txt
```

37. Count how many times each shell is used in **/etc/passwd**.

```
awk -F: '{print $7}' /etc/passwd | sort | uniq -c | sort -nr
```

38. Monitor failed login attempts.

```
grep "Failed password" /var/log/auth.log | awk '{print $(NF-3)}' | sort | uniq -c | sort -nr
```

Advanced Scripting

39. Check which URLs are online.

```
#!/bin/bash
for url in $(cat urls.txt); do
    if curl -s --head "$url" | grep "200 OK" > /dev/null; then
        echo "$url is UP"
    else
        echo "$url is DOWN"
    fi
done
```

40. Simple TODO list manager.

```
#!/bin/bash
case $1 in
    add) echo "$2" >> todo.txt ;;
    list) nl -w2 -s'. ' todo.txt ;;
    remove) sed -i "${2}d" todo.txt ;;
    *) echo "Usage: $0 {add|list|remove} [task]" ;;
esac
```

41. Display system health.

```
echo "CPU Load: $(uptime | awk -F'load average:' '{print $2}')"
echo "Memory: $(free -h | awk '/Mem:/ {print $3 "/" $2}')"
echo "Disk: $(df -h / | awk 'NR==2 {print $5}')"
echo "Uptime: $(uptime -p)"
```

42. Rotate and compress logs.

```bash
#!/bin/bash
log="/var/log/myapp.log"
mv $log $log.$(date +%F)
gzip $log.$(date +%F)
touch $log
```

43. Menu-driven script.

```bash
#!/bin/bash
while true; do
    echo "1. Backup files"
    echo "2. Show processes"
    echo "3. Check memory"
    echo "4. Exit"
    read -p "Choose: " choice

    case $choice in
        1) tar -czf backup.tar.gz /path/to/dir ;;
        2) ps aux ;;
        3) free -h ;;
        4) exit ;;
        *) echo "Invalid option" ;;
    esac
done
```

Networking & Security

44. Ping a list of servers and log results.

```bash
for host in $(cat servers.txt); do
    if ping -c1 $host &>/dev/null; then
        echo "$host is UP"
    else
        echo "$host is DOWN"
    fi
done
```

45. Check open ports.

```bash
netstat -tulnp
# Alternative
ss -tulnp
```

46. Scan local network for active IPs.

```bash
for ip in 192.168.1.{1..254}; do
    ping -c1 -W1 $ip &>/dev/null && echo "$ip is alive"
done
```

```
# Alternative (faster if nmap installed)
nmap -sn 192.168.1.0/24
```

47. Generate random secure passwords.

```
openssl rand -base64 12
# Alternative
pwgen 12 1
# Another way
tr -dc A-Za-z0-9 </dev/urandom | head -c 12
```
