Step 1: Create, Extract, Compress, and Manage tar Backup Archives

1. Command to **extract** the TarDocs.tar archive to the current directory:

   tar -xvvf TarDocs.tar

2. Command to **create** the Javaless_Doc.tar archive from the TarDocs/ directory, while excluding the TarDocs/Documents/Java directory:

   tar -cvvf Javaless_Docs.tar --exclude='TarDocs/Documents/Java' TarDocs/

3. Command to ensure Java/ is not in the new Javaless_Docs.tar archive:

   tar -tvf Javaless_Docs.tar | grep "Java"

Bonus

- Command to create an incremental archive called logs_backup_tar.gz with only changed files to snapshot.file for the /var/log directory: **sudo tar --listed-incremental=snapshot.file -cvzf logs_backup_tar.gz /var/log**

Critical Analysis Question

- Why wouldn't you use the options -x and -c at the same with tar?
- **-x means instructs tar to extract the files from the zipped file**
- **-c means creates a new archive**
- **Not possible to extract something that has not been created, would create an error**

Step 2: Create, Manage, and Automate Cron Jobs

1. Cron job for backing up the /var/log/auth.log file:

   crontab -e

   0 6 * * */3 tar -zcf /auth_backup.tgz /var/log/auth.log

Step 3: Write Basic Bash Scripts

1. Brace expansion command to create the four subdirectories:

   sudo mkdir -p ~/backups/{freemem,diskuse,openlist,freedisk}

Paste your system.sh script edits below:

```
#!/bin/bash
```

2. [Your solution script contents here]

```
  GNU nano 2.9.3                    system.sh                    Modified

#!/bin/bash

# INSTRUCTIONS: Edit the following placeholder command and output $
# For example: cpu_usage_tool > ~/backups/cpuuse/cpu_usage.txt
# The cpu_usage_tool is the command and ~/backups/cpuuse/cpu_usage$
# In the above example, the `cpu_usage_tool` command will output C$
# Do not forget to use the -h option for free memory, disk usage, $

# Free memory output to a free_mem.txt file
free -h > ~/backups/freemem/free_mem.txt

# Disk usage output to a disk_usage.txt file
du -h > ~/backups/diskuse/disk_usage.txt

# List open files to a open_list.txt file
lsof > ~/backups/openlist/open_list.txt

# Free disk space to a free_disk.txt file
df -h > ~/backups/freedisk/free_disk.txt
```

3. Command to make the system.sh script executable:
   chmod +x system.sh

Optional

- Commands to test the script and confirm its execution: **./ sh system.sh**
- **cat ~/backups/freemem/free_mem.txt**

Bonus

- Command to copy system to system-wide cron directory:
- **sudo cp system.sh /etc/cron.weekly**

Step 4. Manage Log File Sizes

1. Run sudo nano /etc/logrotate.conf to edit the logrotate configuration file.

   Configure a log rotation scheme that backs up authentication messages to the /var/log/auth.log.
   - Add your config file edits below:

     sudo nano /etc/logrotate.conf

2. [Your logrotate scheme edits here]

   /var/log/auth.log {

   weekly

   rotate 7

   notifempty

   delaycompress

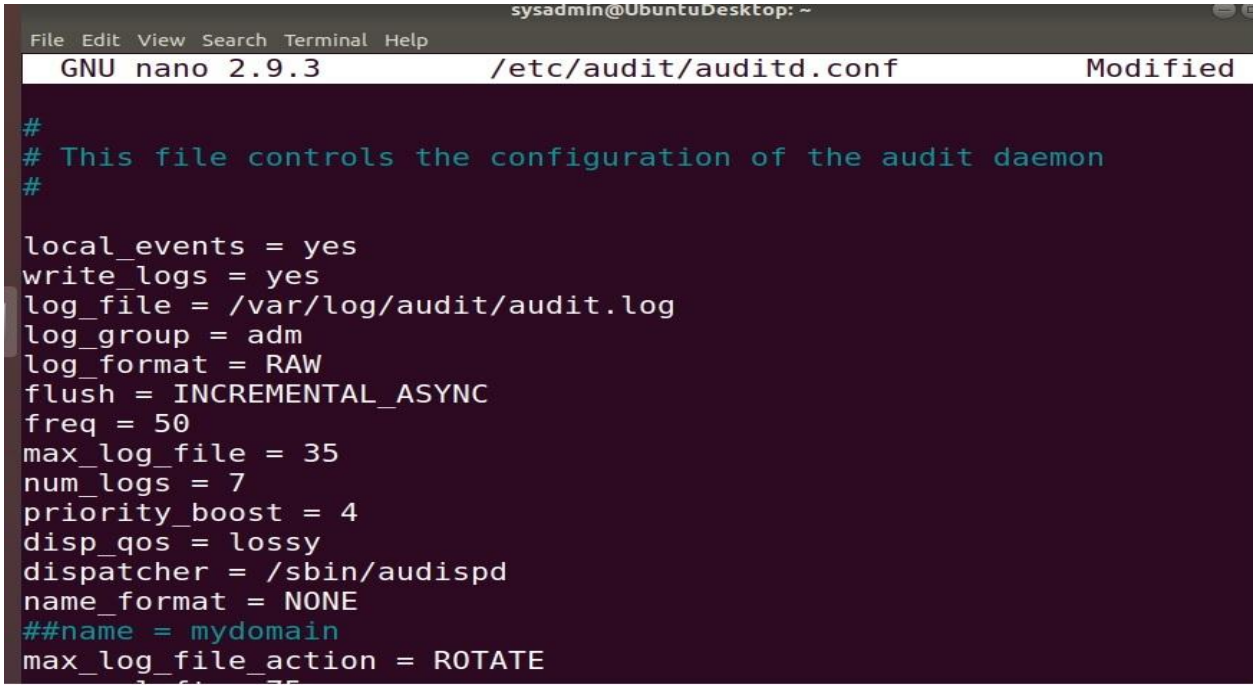
   missingok

   }

Bonus: Check for Policy and File Violations

1. Command to verify auditd is active:
   systemctl status auditd
2. Command to set number of retained logs and maximum log file size: **sudo nano /etc/audit/auditd.conf**

   - Add the edits made to the configuration file below:
3. [Your solution edits here]



4. Command using auditd to set rules for /etc/shadow, /etc/passwd and /var/log/auth.log:

   sudo nano /etc/audit/rules.d/audit.rules

   - Add the edits made to the rules file below:

```
  GNU nano 2.9.3     /etc/audit/rules.d/audit.rules      Modified

## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192

## This determine how long to wait in burst of events
--backlog_wait_time 0

## Set failure mode to syslog
-f 1

#custom rules

-w /etc/shadow -p wa -k shadow
-w /wtc/passwd -p wa -k passwd
-w /etc/shadow -p wra -k hashpass_audit
-w /etc/passwd -p wra -k userpass_audit
-w /var/log/auth.log -p wra -k authlog_audit
```

5. [Your solution edits here]
   Please see screenshot above
6. Command to restart auditd:
   sudo systemctl restart auditd
7. Command to list all auditd rules:
   sudo auditctl -l
8. Command to produce an audit report:
   sudo aureport -au
9. Create a user with sudo useradd attacker and produce an audit report that lists account modifications: **sudo adduser attacker**

   sudo aureport -au

10. Command to use auditd to watch /var/log/cron:
    sudo auditctl -w /var/log/cron
11. Command to verify auditd rules:
    sudo auditctl -l

Bonus (Research Activity): Perform Various Log Filtering Techniques

1. Command to return journalctl messages with priorities from emergency to error: **sudo journalctl -b -p emerg..err**

2. Command to check the disk usage of the system journal unit since the most recent boot: **sudo journalctl --disk-usage --boot=-0**

3. Command to remove all archived journal files except the most recent two: **sudo journalctl --vaccum-file=2**

4. Command to filter all log messages with priority levels between zero and two, and save output to /home/sysadmin/Priority_High.txt: **sudo journalctl -p 0..2 > /home/sysadmin/Priority_High.txt**

5. Command to automate the last command in a daily cronjob. Add the edits made to the crontab file below: **sudo crontab -e**

   [Your solution cron edits here]