D796 - Task 1 - Attempt 1

Silver Alcid - #011933973

9/10/2025

Linux & Unix

A. Create a shell script to create a new user called create_user.sh

create_user.sh

```
#!/usr/bin/env bash
# Create a new Linux user and ensure group membership.
# Usage: sudo ./create_user.sh <username>

set -euo pipefail

# --- 1) Validate input ---
if [[ $# -lt 1 ]]; then
  echo "Error: Missing username argument." >&2
  echo "Usage: sudo $0 <username>" >&2
  exit 1
fi

if [[ $EUID -ne 0 ]]; then
  echo "Error: This script must be run as root (use sudo)." >&2
  exit 1
fi

USERNAME="$1"
GROUP="dev_group"
HARDCODED_PASS="password"

# --- 2) Ensure group exists ---
if ! getent group "$GROUP" >/dev/null 2>&1; then
  groupadd "$GROUP"
  echo "Created group: $GROUP"
else
  echo "Group already exists: $GROUP"
fi

# --- 3) Add user and assign password ---
if id -u "$USERNAME" >/dev/null 2>&1; then
  echo "Error: User '$USERNAME' already exists." >&2
  exit 1
fi

# create user with home directory and primary group
useradd -m -g "$GROUP" -s /bin/bash "$USERNAME"

# set hardcoded password
echo "${USERNAME}:${HARDCODED_PASS}" | chpasswd

# force password change at next login
chage -d 0 "$USERNAME"

echo "User '$USERNAME' created with initial password: ${HARDCODED_PASS}"

# --- 4) Display /etc/passwd to verify user creation ---
echo
echo "===== /etc/passwd ====="
cat /etc/passwd
echo "========================"

```

B. Create a shell script to delete the user from part A called delete_user.sh

delete_user.sh

```bash
#!/usr/bin/env bash
# Delete a Linux user and their home directory, with confirmation and verification.

set -euo pipefail

# Must be run as root
if [[ "${EUID}" -ne 0 ]]; then
  echo "Error: This script must be run as root." >&2
  exit 1
fi

# 1) Verify username argument provided
if [[ $# -lt 1 || -z "${1:-}" ]]; then
  echo "Usage: $0 <username>" >&2
  exit 1
fi

USER_TO_DELETE="$1"

# Verify that user exists before proceeding
if ! id -u "$USER_TO_DELETE" >/dev/null 2>&1; then
  echo "Error: User '$USER_TO_DELETE' does not exist." >&2
  exit 1
fi

# 2) Ask for confirmation
read -r -p "Are you sure you want to delete user '$USER_TO_DELETE' and their home directory? [y/N]: " CONFIRM
case "$CONFIRM" in
  y|Y|yes|YES) ;;
  *) echo "Aborted."; exit 0 ;;
esac

# 3) Delete the user and home directory
#    (userdel -r removes the user's home and mail spool)
if userdel -r "$USER_TO_DELETE"; then
  echo "User '$USER_TO_DELETE' deleted."
else
  echo "Error: Failed to delete user '$USER_TO_DELETE'." >&2
  exit 1
fi

# 4) Display /etc/passwd to verify deletion
echo "===== /etc/passwd (post-deletion) ====="
cat /etc/passwd

```

C. Configure the shell initialization files to customize the environment for each new login or interactive shell session after login

```
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi
# Colors: user (green), host (cyan), cwd (magenta), prompt symbol (yellow)
PS1="\[\e[1;32m\]\u\[\e[0m\]@\[\e[36m\]\h \[\e[35m\]\w\[\e[0m\]\n\[\e[33m\]\\$\[\e[0m\] "
export PS1


# ----- Custom prompt -----
# Colors: user (green), host (cyan), cwd (magenta), prompt symbol (yellow)
PS1="\[\e[1;32m\]\u\[\e[0m\]@\[\e[36m\]\h \[\e[35m\]\w\[\e[0m\]\n\[\e[33m\]\\$\[\e[0m\] "
export PS1
# -------------------------

# Load custom aliases
if [ -f ~/.bash_aliases ]; then
  . ~/.bash_aliases
fi

# Add ~/bin to PATH if it exists
if [ -d "$HOME/bin" ]; then
  PATH="$HOME/bin:$PATH"
fi
export PATH
```

home > silveralcid > .bash_aliases

```
1   # Listing shortcuts
2   alias ll='ls -lrt'
3   alias la='ls -a'
4   alias c='clear'
5
6   # Quick navigation shortcuts
7   alias dsk='cd "$HOME/Desktop"'
8   alias dld='cd "$HOME/Downloads"'
9   alias doc='cd "$HOME/Documents"'
10
```

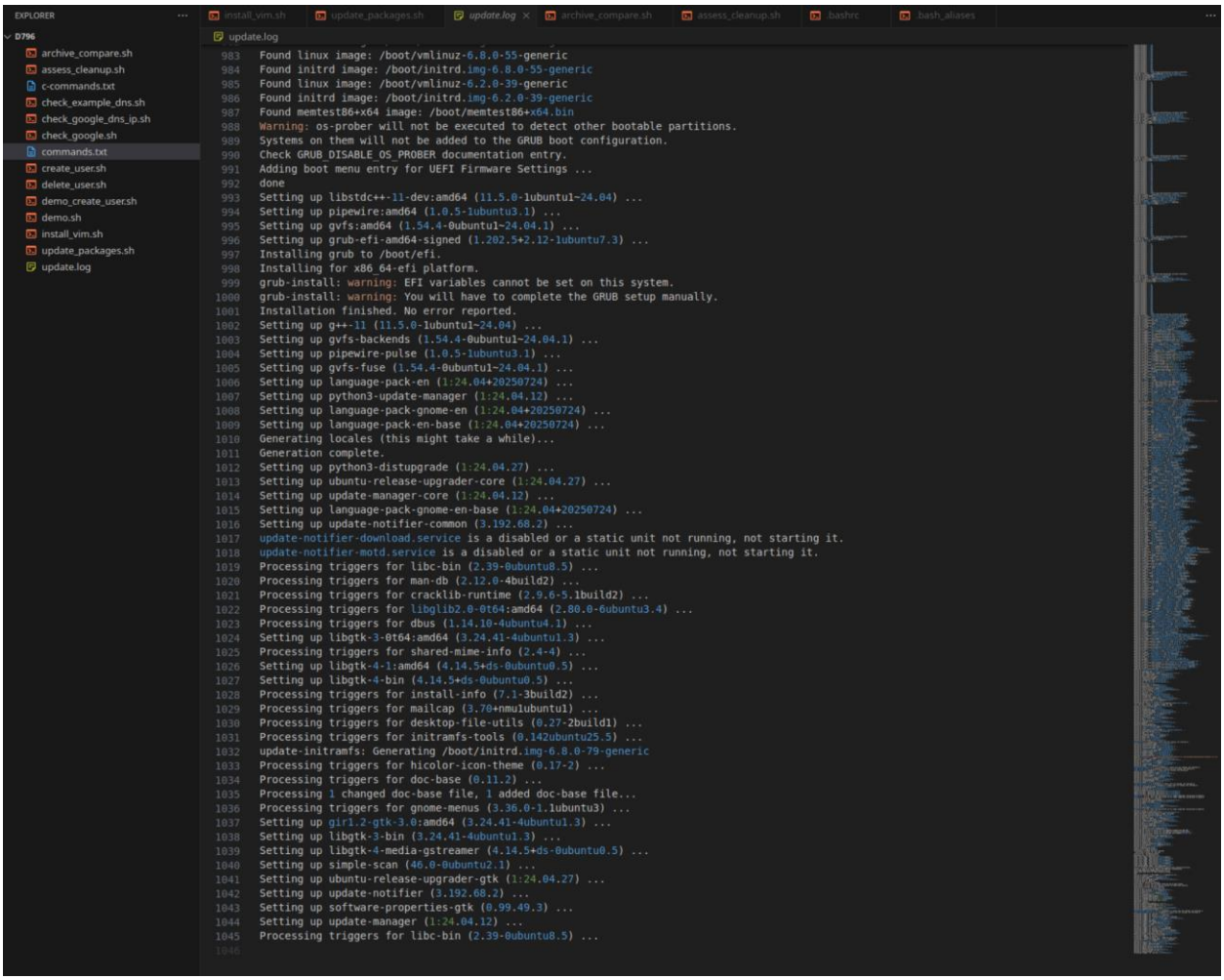
D. Add and update the packages

install_vim.sh

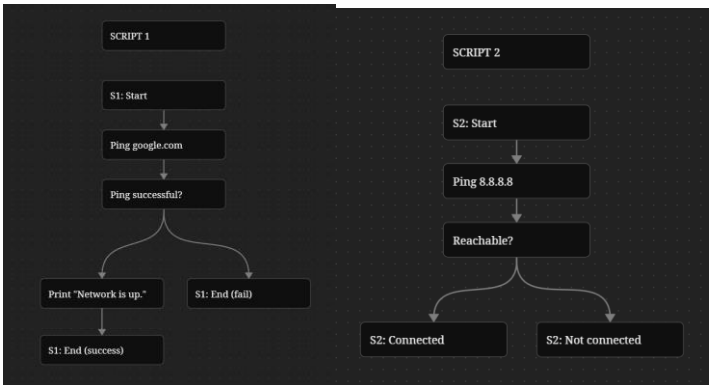
```
1   #!/bin/bash
2
3   # Check if vim is installed
4   if command -v vim >/dev/null 2>&1; then
5       echo "Vim is already installed"
6   else
7       echo "Installing Vim..."
8       # Update package list and install vim
9       sudo apt-get update -y
10      sudo apt-get install vim -y
11  fi
12
```

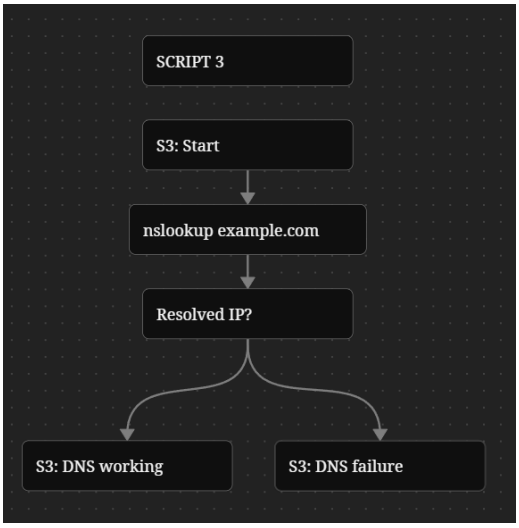
update_packages.sh

```
1
2   sudo apt-get update -y > update.log 2>&1
3
4   sudo apt-get upgrade -y >> update.log 2>&1
5
6   echo "System update complete. See update.log for details."
7
```



E. Create three shell scripts that check network connections





F. Create a shell script that will assess and clean up disk space

commands.txt | assess_cleanup.sh

assess_cleanup.sh

```bash
#!/usr/bin/env bash

set -euo pipefail

TARGET_USER="silveralcid"
USER_HOME="$(getent passwd "$TARGET_USER" | cut -d: -f6)"
: "${USER_HOME:?Error: could not resolve home for $TARGET_USER}"

# 1. Get free disk space in root partition
initial_free=$(df --output=avail / | tail -n 1)

# 2. Function to delete contents of a directory
cleanDir() {
  local dir="$1"
  if [[ -d "$dir" ]]; then
    rm -rf "${dir:?}/"* "${dir:?}"/.[!.]* "${dir:?}"/..?* 2>/dev/null || true
    echo "Cleaned: $dir"
  else
    echo "Directory not found: $dir"
  fi
}

# 3. List of directories to clean
dirs_to_clean=(
  "/var/log"
  "$USER_HOME/.cache"
  "$USER_HOME/tmp"
)

# 4. Delete files and subdirectories in specified directories
for dir in "${dirs_to_clean[@]}"; do
  cleanDir "$dir"
done

# 5. Get free disk space after cleanup
final_free=$(df --output=avail / | tail -n 1)

# Compare disk space changes
freed=$((final_free - initial_free))

if ((freed > 0)); then
  echo "Disk space freed: $freed KB"
else
  echo "No significant disk space was freed"
fi

```

G. Create a shell script that archives and compresses files and directories

commands.txt | archive_compare.sh

archive_compare.sh

```bash
#!/usr/bin/env bash

set -euo pipefail

# Config
TARGET_USER="silveralcid"
# Output under target user's home
USER_HOME="$(getent passwd "$TARGET_USER" | cut -d: -f6)"
: "${USER_HOME:?Error: could not resolve home for $TARGET_USER}"
OUTDIR="${USER_HOME}/backups/etc"

# If not root, re-exec with sudo
if [[ ${EUID:-$(id -u)} -ne 0 ]]; then
  exec sudo --preserve-env=TARGET_USER,OUTDIR,USER_HOME,BASHOPTS,BASH_XTRACEFD "$0" "$@"
fi

# Ensure output dir exists and is owned by target user
install -d -m 0750 -o "$TARGET_USER" -g "$(id -gn "$TARGET_USER")" "$OUTDIR"

# fileSize(): print file size in bytes
fileSize() {
  local f="${1:-}"
  if [[ -z "$f" ]]; then
    echo "Error: fileSize() missing file argument" >&2
    return 2
  fi
  if [[ ! -f "$f" ]]; then
    echo "Error: file not found: $f" >&2
    return 2
  fi

  # Prefer GNU stat; fall back to wc -c for portability
  if stat --version >/dev/null 2>&1; then
    stat -c '%s' -- "$f"
  else
    wc -c <"$f" | tr -d ' '
  fi
}

timestamp="$(date +%Y%m%d-%H%M%S)"
base_gz="$OUTDIR/etc-$timestamp.tar.gz"
base_bz2="$OUTDIR/etc-$timestamp.tar.bz2"

# Create archives (root needed to read all of /etc)
tar -C / -czpf "$base_gz" etc
tar -C / -cjpf "$base_bz2" etc

```

```
47
48  # Calculate sizes
49  size_gz="$(fileSize "$base_gz")"
50  size_bz2="$(fileSize "$base_bz2")"
51
52  # Optional human-readable formatter
53  hr() { command -v numfmt >/dev/null 2>&1 && numfmt --to=iec --suffix=B "$1" || echo "$1 B"; }
54
55  # Report results and difference
56  diff_abs=$(( size_gz - size_bz2 ))
57  (( diff_abs < 0 )) && diff_abs=$(( -diff_abs ))
58
59  echo "Created:"
60  echo "  GZIP : $base_gz  ($(hr "$size_gz"))"
61  echo "  BZIP2: $base_bz2 ($(hr "$size_bz2"))"
62
63  if (( size_gz < size_bz2 )); then
64    echo "Result: gzip archive is smaller by $(hr "$diff_abs")."
65  elif (( size_bz2 < size_gz )); then
66    echo "Result: bzip2 archive is smaller by $(hr "$diff_abs")."
67  else
68    echo "Result: both archives are identical in size."
69  fi
70
71  # Ensure the target user can access the outputs
72  chown "$TARGET_USER":"$(id -gn "$TARGET_USER")" "$base_gz" "$base_bz2"
73  chmod 0640 "$base_gz" "$base_bz2"
74
```

H. Record yourself completing each task using Panopto with a clear view of yourself and the screen

https://wgu.hosted.panopto.com/Panopto/Pages/Viewer.aspx?id=998ebdfe-e640-448f-9df3-b3540174aadb

I. Acknowledge sources, using APA-formatted in-text citations and references, for content that is quoted, paraphrased, or summarized.

No sources were needed or used.