

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
1  #!/usr/bin/env bash
2  # Create a new Linux user and ensure group membership.
3  # Usage: sudo ./create_user.sh <username>
4
5  set -euo pipefail
6
7  # --- 1) Validate input ---
8  if [[ $# -lt 1 ]]; then
9      echo "Error: Missing username argument." >&2
10     echo "Usage: sudo $0 <username>" >&2
11     exit 1
12 fi
13
14 if [[ $EUID -ne 0 ]]; then
15     echo "Error: This script must be run as root (use sudo)." >&2
16     exit 1
17 fi
18
19 USERNAME="$1"
20 GROUP="dev_group"
21 HARDCODED_PASS="password"
22
23 # --- 2) Ensure group exists ---
24 if ! getent group "$GROUP" >/dev/null 2>&1; then
25     groupadd "$GROUP"
26     echo "Created group: $GROUP"
27 else
28     echo "Group already exists: $GROUP"
29 fi
30
31 # --- 3) Add user and assign password ---
32 if id -u "$USERNAME" >/dev/null 2>&1; then
33     echo "Error: User '$USERNAME' already exists." >&2
34     exit 1
35 fi
36
37 # create user with home directory and primary group
38 useradd -m -g "$GROUP" -s /bin/bash "$USERNAME"
39
40 # set hardcoded password
41 echo "${USERNAME}:${HARDCODED_PASS}" | chpasswd
42
43 # force password change at next login
44 chage -d 0 "$USERNAME"
45
46 echo "User '$USERNAME' created with initial password: ${HARDCODED_PASS}"
47
48 # --- 4) Display /etc/passwd to verify user creation ---
49 echo
50 echo "===== /etc/passwd ====="
51 cat /etc/passwd
52 echo "===== "
53

```

## B. Create a shell script to delete the user from part A called delete\_user.sh

```

delete_user.sh
1  #!/usr/bin/env bash
2  # Delete a Linux user and their home directory, with confirmation and verification.
3
4  set -euo pipefail
5
6  # Must be run as root
7  if [[ "${EUID}" -ne 0 ]]; then
8      echo "Error: This script must be run as root." >&2
9      exit 1
10 fi
11
12 # 1) Verify username argument provided
13 if [[ $# -lt 1 || -z "${1:-}" ]]; then
14     echo "Usage: $0 <username>" >&2
15     exit 1
16 fi
17
18 USER_TO_DELETE="$1"
19
20 # Verify that user exists before proceeding
21 if ! id -u "$USER_TO_DELETE" >/dev/null 2>&1; then
22     echo "Error: User '$USER_TO_DELETE' does not exist." >&2
23     exit 1
24 fi
25
26 # 2) Ask for confirmation
27 read -r -p "Are you sure you want to delete user '$USER_TO_DELETE' and their home directory? [y/N]: " CONFIRM
28 case "$CONFIRM" in
29     y|Y|yes|YES) ;;
30     *) echo "Aborted."; exit 0 ;;
31 esac
32
33 # 3) Delete the user and home directory
34 # (userdel -r removes the user's home and mail spool)
35 if userdel -r "$USER_TO_DELETE"; then
36     echo "User '$USER_TO_DELETE' deleted."
37 else
38     echo "Error: Failed to delete user '$USER_TO_DELETE'." >&2
39     exit 1
40 fi
41
42 # 4) Display /etc/passwd to verify deletion
43 echo "==== /etc/passwd (post-deletion) ====="
44 cat /etc/passwd
45

```

## 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

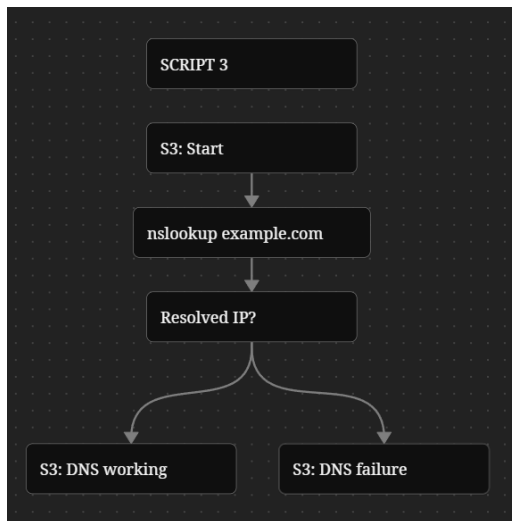
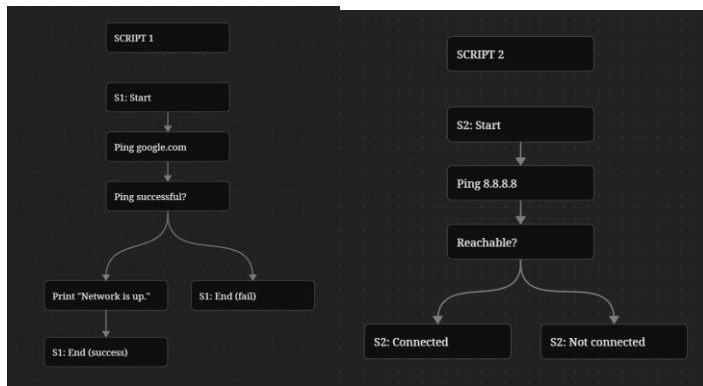
```

```

update.log
983 Found linux image: /boot/vmlinuz-6.8.0-55-generic
984 Found initrd image: /boot/initrd.img-6.8.0-55-generic
985 Found linux image: /boot/vmlinuz-6.2.0-39-generic
986 Found initrd image: /boot/initrd.img-6.2.0-39-generic
987 Found memtest86+x64 image: /boot/memtest86+x64.bin
988 Warning: os-prober will not be executed to detect other bootable partitions.
989 Systems on them will not be added to the GRUB boot configuration.
990 Check GRUB_DISABLE_OS_PROBER documentation entry.
991 Adding boot menu entry for UEFI Firmware Settings ...
992 done
993 Setting up libstdc++-11-dev:amd64 (11.5.0-1ubuntu24.04) ...
994 Setting up pipewire:amd64 (1.0.5-1ubuntu3.1) ...
995 Setting up gvfs:amd64 (1.54.4-0ubuntu1-24.04.1) ...
996 Setting up grub-efi-amd64-signed (1.202.5+2.12-1ubuntu7.3) ...
997 Installing grub to /boot/efi.
998 Installing for x86_64-efi platform.
999 grub-install: warning: EFI variables cannot be set on this system.
1000 grub-install: warning: You will have to complete the GRUB setup manually.
1001 Installation finished. No error reported.
1002 Setting up g++-11 (11.5.0-1ubuntu24.04) ...
1003 Setting up gvfs-backends (1.54.4-0ubuntu1-24.04.1) ...
1004 Setting up pipewire-pulse (1.0.5-1ubuntu3.1) ...
1005 Setting up gvfs-fuse (1.54.4-0ubuntu1-24.04.1) ...
1006 Setting up language-pack-en (1:24.04+20250724) ...
1007 Setting up python3-update-manager (1:24.04.12) ...
1008 Setting up language-pack-gnome-en (1:24.04+20250724) ...
1009 Setting up language-pack-en-base (1:24.04+20250724) ...
1010 Generating locales (this might take a while)...
1011 Generation complete.
1012 Setting up python3-distupgrade (1:24.04.27) ...
1013 Setting up ubuntu-release-upgrader-core (1:24.04.27) ...
1014 Setting up update-manager-core (1:24.04.12) ...
1015 Setting up language-pack-gnome-en-base (1:24.04+20250724) ...
1016 Setting up update-notifier-common (3.192.68.2) ...
1017 update-notifier-download.service is a disabled or a static unit not running, not starting it.
1018 update-notifier-motd.service is a disabled or a static unit not running, not starting it.
1019 Processing triggers for libc-bin (2.39-0ubuntu0.5) ...
1020 Processing triggers for man-db (2.12.0-4build2) ...
1021 Processing triggers for cracklib-runtime (2.9.6-5.1build2) ...
1022 Processing triggers for libgl1:amd64 (2.80.0-6ubuntu3.4) ...
1023 Processing triggers for dbus (1.14.10-0ubuntu4.1) ...
1024 Setting up libgtk-3-0t64:amd64 (3.24.41-4ubuntu1.3) ...
1025 Processing triggers for shared-mime-info (2.4-4) ...
1026 Setting up libgtk-4-1:amd64 (4.14.5+ds-0ubuntu0.5) ...
1027 Setting up libgtk-4-bin (4.14.5+ds-0ubuntu0.5) ...
1028 Processing triggers for install-info (7.1-3build2) ...
1029 Processing triggers for mailcap (3.70+nmu1ubuntu1) ...
1030 Processing triggers for desktop-file-utils (0.27-2build1) ...
1031 Processing triggers for initramfs-tools (0.142ubuntu25.5) ...
1032 update-initramfs: Generating /boot/initrd.img-6.8.0-79-generic
1033 Processing triggers for hicolor-icon-theme (0.17-2) ...
1034 Processing triggers for doc-base (0.11.2) ...
1035 Processing 1 changed doc-base file, 1 added doc-base file...
1036 Processing triggers for gnome-menus (3.36.0-1ubuntu3) ...
1037 Setting up gir1.2-gtk-3.0:amd64 (3.24.41-4ubuntu1.3) ...
1038 Setting up libgtk-3-bin (3.24.41-4ubuntu1.3) ...
1039 Setting up libgtk-4-media-gstreamer (4.14.5+ds-0ubuntu0.5) ...
1040 Setting up simple-scan (46.0-0ubuntu2.1) ...
1041 Setting up ubuntu-release-upgrader-gtk (1:24.04.27) ...
1042 Setting up update-notifier (3.192.68.2) ...
1043 Setting up software-properties-gtk (0.99.49.3) ...
1044 Setting up update-manager (1:24.04.12) ...
1045 Processing triggers for libc-bin (2.39-0ubuntu0.5) ...
1046

```

**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 x
assess_cleanup.sh
1  #!/usr/bin/env bash
2
3  set -euo pipefail
4
5  TARGET_USER="silveralcid"
6  USER_HOME="$(getent passwd "$TARGET_USER" | cut -d: -f6)"
7  : "${USER_HOME:?Error: could not resolve home for $TARGET_USER}"
8
9  # 1. Get free disk space in root partition
10 initial_free=$(df --output=avail / | tail -n 1)
11
12 # 2. Function to delete contents of a directory
13 cleanDir() {
14     local dir="$1"
15     if [[ -d "$dir" ]]; then
16         rm -rf "${dir:?}/${dir:?}/.*" "${dir:?}/.?!.*" "${dir:?}/..?* 2>/dev/null || true
17         echo "Cleaned: $dir"
18     else
19         echo "Directory not found: $dir"
20     fi
21 }
22
23 # 3. List of directories to clean
24 dirs_to_clean=(
25     "/var/log"
26     "$USER_HOME/.cache"
27     "$USER_HOME/tmp"
28 )
29
30 # 4. Delete files and subdirectories in specified directories
31 for dir in "${dirs_to_clean[@]}; do
32     cleanDir "$dir"
33 done
34
35 # 5. Get free disk space after cleanup
36 final_free=$(df --output=avail / | tail -n 1)
37
38 # Compare disk space changes
39 freed=$((final_free - initial_free))
40
41 if ((freed > 0)); then
42     echo "Disk space freed: $freed KB"
43 else
44     echo "No significant disk space was freed"
45 fi
46

```

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



```

commands.txt  archive_compare.sh x
archive_compare.sh
1  #!/usr/bin/env bash
2
3  set -euo pipefail
4
5  # Config
6  TARGET_USER="silveralcid"
7  # Output under target user's home
8  USER_HOME="$(getent passwd "$TARGET_USER" | cut -d: -f6)"
9  : "${USER_HOME:?Error: could not resolve home for $TARGET_USER}"
10 OUTDIR="${USER_HOME}/backups/etc"
11
12 # If not root, re-exec with sudo
13 if [[ ${EUID:-$(id -u)} -ne 0 ]]; then
14     exec sudo --preserve-env=TARGET_USER,OUTDIR,USER_HOME,BASHOPTS,BASH_XTRACEFD "$0" "$@"
15 fi
16
17 # Ensure output dir exists and is owned by target user
18 install -d -m 0750 -o "$TARGET_USER" -g "$(id -gn "$TARGET_USER")" "$OUTDIR"
19
20 # fileSize(): print file size in bytes
21 fileSize() {
22     local f="${1:-}"
23     if [[ -z "$f" ]]; then
24         echo "Error: fileSize() missing file argument" >&2
25         return 2
26     fi
27     if [[ ! -f "$f" ]]; then
28         echo "Error: file not found: $f" >&2
29         return 2
30     fi
31
32     # Prefer GNU stat; fall back to wc -c for portability
33     if stat --version >/dev/null 2>&1; then
34         stat -c '%s' -- "$f"
35     else
36         wc -c <"$f" | tr -d ' '
37     fi
38 }
39
40 timestamp="$(date +%Y%m%d-%H%M%S)"
41 base_gz="$OUTDIR/etc-$timestamp.tar.gz"
42 base_bz2="$OUTDIR/etc-$timestamp.tar.bz2"
43
44 # Create archives (root needed to read all of /etc)
45 tar -C / -czpf "$base_gz" etc
46 tar -C / -cjpz "$base_bz2" etc
47

```



```

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.

