Examen Practico Alejandro Almagro Torregrosa Segunda Evaluación

1. Completa tu propuesta de BASH con las opciones que consideres y / o las que se comentaron en la TI de corrección de la misma.

Como comentamos en la ti al script le falta un poco de interactividad con el usuario y lo que viene a ser siendo manejo de errores como tal o problemas.

Empezemos por los archivos de mysql, visto lo cual tener que escribir a mano las variables es un poco tedioso por tanto vamos a ir mejorando el script.

Antes:

D: > Descargas > $ usuario.sh

```
# Ejecuta los comandos SQL en MySQL
mysql -u "$MYSQL_ROOT_USER" <<EOF
-- Elimina el usuario "acore" en cualquier host, si existe.
DROP USER IF EXISTS '$NEW_USER'@'%';
DROP USER IF EXISTS '$NEW_USER'@'localhost';

-- Crea el usuario "acore" permitiendo conexiones desde cualquier host (%).
CREATE USER '$NEW_USER'@'%' IDENTIFIED BY '$NEW_USER_PASS';

-- Otorga todos los privilegios sobre todas las bases de datos y la opción de otorgar privilegios a otros.
GRANT ALL PRIVILEGES ON *.* TO '$NEW_USER'@'%' WITH GRANT OPTION;

-- Aplica los cambios
FLUSH PRIVILEGES;
EOF
```

Despues de la mejora:

-Se aplica una gestion de errores en los parametros que introduzca el usuario:

```
#!/bin/bash

# Solicita el nombre de usuario
read -p "Ingrese el nombre del usuario a crear: " NEW_USER

# Verifica que no esté vacío, no tenga simbolos o numeros
if [[ -z "$NEW_USER" || "$NEW_USER" =~ [^a-zA-Z] ]]; then
    echo "Error: El nombre de usuario no puede estar vacío y debe contener solo letras."
    exit 1
fi

# Solicita la contraseña sin mostrarla en pantalla
read -s -p "Ingrese la contraseña para $NEW_USER: " NEW_USER_PASS
echo

# Verifica que no esté vacía
if [[ -z "$NEW_USER_PASS" ]]; then
    echo "Error: La contraseña no puede estar vacía."
    exit 1
```

Se deja el resto del codigo igual, añadiendo una gestion de errores por si falla en mysql:

```
# Ejecuta los comandos SQL en MySQL y captura errores
mysql -u "$MYSQL_ROOT_USER" <<EOF
-- Elimina el usuario si ya existe
DROP USER IF EXISTS '$NEW_USER'@'%';
DROP USER IF EXISTS '$NEW_USER'@'localhost';

-- Crea el usuario con la contraseña ingresada
CREATE USER '$NEW_USER'@'%' IDENTIFIED BY '$NEW_USER_PASS';

-- Otorga permisos completos
GRANT ALL PRIVILEGES ON *.* TO '$NEW_USER'@'%' WITH GRANT OPTION;

-- Aplica los cambios
FLUSH PRIVILEGES;
EOF

# Verifica si el comando anterior tuvo éxito
if [ $? -eq 0 ]; then
    echo "Usuario $NEW_USER creado y configurado correctamente en MySQL."
else
    echo "Error: No se pudo crear el usuario $NEW_USER en MySQL. Verifique los datos ingresados y los permisos del usu
    exit 1
fi
```

Ahora pasemos con el siguiente script que daba nombre al realm y hagamoslo parecido:

Gestion de errores antes en los parametros que tiene que introducir:

```
pts > $ realm.sh
#!/bin/bash
# actualizar_realmlist.sh
# Este script actualiza la dirección y el nombre del realmlist en la base de datos acore_auth.

# Solicita la nueva dirección del realmlist
read -p "Ingrese la nueva dirección del realmlist: " NUEVA_DIRECCION

# Verifica que no esté vacía y que sea una dirección IP válida
if [[ -z "$NUEVA_DIRECCION" ]]; then
    echo "Error: La dirección IP no puede estar vacía."
    exit 1
elif ! [[ "$NUEVA_DIRECCION" =~ ^[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+$ ]]; then
    echo "Error: La dirección IP no es válida."
    exit 1
fi

# Solicita el nuevo nombre del realmlist
read -p "Ingrese el nuevo nombre del realmlist: " NUEVO_NOMBRE

# Verifica que no esté vacío y que solo contenga letras
if [[ -z "$NUEVO_NOMBRE" ]]; then
    echo "Error: El nombre del realmlist no puede estar vacío."
    exit 1
elif ! [[ "$NUEVO_NOMBRE" =~ ^[a-zA-Z]+$ ]]; then
    echo "Error: El nombre del realmlist solo puede contener letras."
    exit 1
fi
```

Ahora tenemos la gestion de errores posteriores:

```
# Define la base de datos y el usuario root de MySQL
MYSQL_ROOT_USER="root"
MYSQL_DB="acore_auth"

# Solicita la contraseña de MySQL de forma segura
read -s -p "Ingrese la contraseña de MySQL: " MYSQL_ROOT_PASS
echo

# Ejecuta los comandos SQL para actualizar el realmlist
mysql -u "$MYSQL_ROOT_USER" -p"$MYSQL_ROOT_PASS" "$MYSQL_DB" <<EOF
UPDATE realmlist SET address = '$NUEVA_DIRECCION' WHERE id = 1;
UPDATE realmlist SET name = '$NUEVO_NOMBRE' WHERE id = 1;
FLUSH PRIVILEGES;
EOF
if [ $? -eq 0 ]; then
    echo "El realmlist ha sido actualizado: dirección '$NUEVA_DIRECCION' y nombre '$NUEVO_NOMBRE'."
else
    echo "Error: No se pudo actualizar el realmlist. Por favor, intente nuevamente."
    exit 1
fi
```

Ahora pasemos al cuerpo principal del script primer.sh, al anterior script le faltaba un manejo de errores por si el usuario introducia o marcaba cosas que no son, ademas de interactuar con el para pedirle datos o confirmaciones en pasos importantes, vamos a ir paso por paso lo que hemos modificado:

Se mantienen algunos de los mensajes de estado:

```
# Mensajes de estado
confirmacion="Directorio Creado"
actualizacion="Equipo Actualizado"
directorio="Dir Creado"
git="Git Clonado"
deps="Dependencias Instaladas"
compilado="Compilación Finalizada"
cliente="Descarga del cliente finalizada"
archivos="Configuraciones copiadas"
```

Ahora cuando actualizemos el sistema lo haremos de esta manera:

```
# ==============================
# 🚀 ACTUALIZACIÓN DEL SISTEMA
# ==============================
echo "¿Deseas actualizar el sistema? (s/n)"
read -r respuesta
if [[ "$respuesta" == "s" || "$respuesta" == "S" ]]; then
    apt update && apt upgrade -y && apt install git -y
    echo "$actualizacion"
else
    echo "Omitiendo actualización..."
fi
```

Ahora con la creacionde directorio haremos algo parecido:

```
# ==============================
# 📂 CREACIÓN DE DIRECTORIOS
# ==============================
# Solicitar el nombre del directorio
read -p "Ingrese el nombre del directorio donde se instalará el servidor: " DIR_NAME

# Si el usuario no ingresa nada, usar "Serverwow" por defecto
DIR_NAME=${DIR_NAME:-Serverwow}

# Verificar si el directorio existe
if [[ ! -d "$DIR_NAME" ]]; then
    mkdir "$DIR_NAME"
    echo "Directorio '$DIR_NAME' creado."
else
    echo "El directorio '$DIR_NAME' ya existe."
fi

# Entrar en el directorio o salir si falla
cd "$DIR_NAME" || { echo "Error: No se pudo acceder al directorio '$DIR_NAME'."; exit 1; }
```

Cuando clonemos el repo si pasa algo tendremos el aviso de que ya lo hemos clonado o de que no podemos acceder al directorio:

```
# ==============================
# 🛠️ CLONACIÓN DEL REPO
# ==============================
if [[ ! -d "azerothcore-wotlk" ]]; then
    git clone https://github.com/azerothcore/azerothcore-wotlk.git
    echo "$git"
else
    echo "El repositorio ya está clonado."
fi
cd azerothcore-wotlk || { echo "Error: No se pudo entrar en azerothcore-wotlk"; exit 1; }
```

Instalacion, Compilacion y Descarga del Cliente:

```
# ==============================
# 📌 INSTALAR DEPENDENCIAS
# ==============================
echo "¿Deseas instalar dependencias? (s/n)"
read -r respuesta
if [[ "$respuesta" == "s" || "$respuesta" == "S" ]]; then
    ./acore.sh install-deps
    echo "$deps"
else
    echo "Omitiendo instalación de dependencias..."
fi

# ==============================
# 🔨 COMPILACIÓN DEL SERVIDOR
# ==============================
echo "¿Deseas compilar AzerothCore? (s/n)"
read -r respuesta
if [[ "$respuesta" == "s" || "$respuesta" == "S" ]]; then
    ./acore.sh compiler all
    echo "$compilado"
else
    echo "Omitiendo compilación..."
fi

# ==============================
# 🎮 DESCARGA DEL CLIENTE
# ==============================
echo "¿Deseas descargar los datos del cliente? (s/n)"
read -r respuesta
if [[ "$respuesta" == "s" || "$respuesta" == "S" ]]; then
    ./acore.sh client-data
    echo "$cliente"
else
    echo "Omitiendo descarga del cliente..."
fi
```

Ahora copiemos los archivos de configuracion

```
# ==============================
# ⚙️ COPIAR ARCHIVOS DE CONFIGURACIÓN
# ==============================
if cp env/dist/etc/authserver.conf.dist env/dist/etc/authserver.conf; then
    echo "authserver.conf copiado correctamente."
else
    echo "Error al copiar authserver.conf."
    exit 1
fi

if cp env/dist/etc/worldserver.conf.dist env/dist/etc/worldserver.conf; then
    echo "worldserver.conf copiado correctamente."

else
    echo "Error al copiar worldserver.conf."
    exit 1

fi

echo "$archivos"
```

Y por ultimo se le añaden los dos scripts de mysql que necesitas redirigir a tu directorio:

```
while true; do
    read -p "Ingrese el nombre del directorio donde tienes los scripts (o 'exit' para salir): " SCRIPT_DIR
    if [[ "$SCRIPT_DIR" == "exit" ]]; then
        echo "Saliendo del proceso..."
        exit 0
    elif cd "$SCRIPT_DIR"; then
        echo "✓ Accedido al directorio '$SCRIPT_DIR' correctamente."
        break
    else
        echo "❌ Error: No se pudo acceder al directorio '$SCRIPT_DIR'. Inténtalo de nuevo."
    fi
done
# ==============================
# 🔑 CREACIÓN DEL USUARIO MYSQL
# ==============================
echo "🔑 Creando usuario MySQL..."
./usuario.sh

# ==============================
# 🛠️ CONFIGURACIÓN DEL REALMLIST
# ==============================
echo "🛠️ Actualizando realmlist..."
./realm.sh
```

Asi no dara error porque no encuentra los archivos:

Mejora del ultimo script start_esp, aqui verificamos si tmux esta instalada, si las versiones ya existen en el ordenador por si has borrado la carpeta para reinstalar o otra cosa, evitamos la ejecucion duplicada de sesiones, y daremos unos mensajes mas claros al usuario.

```
#!/bin/bash
AUTHSERVER_CMD="~/azerothcore-wotlk/acore.sh run-authserver"
WORLDSERVER_CMD="~/azerothcore-wotlk/acore.sh run-worldserver"
AUTHSESSION="auth-session"
WORLDSESSION="world-session"

# ============================
# VERIFICAR QUE TMUX ESTÉ INSTALADO
# ============================
if ! command -v tmux &> /dev/null; then
    echo "Error: tmux no está instalado. Instálalo con: sudo apt install tmux"
    exit 1
fi
```

Creacion de sesiones:

```
# ============================
# CREAR SESIÓN DE AUTHSERVER
# ============================
if tmux has-session -t $AUTHSESSION 2>/dev/null; then
    echo "La sesión $AUTHSESSION ya está en ejecución."
else
    tmux new-session -d -s $AUTHSESSION
    echo "Sesión de authserver creada: $AUTHSESSION"
    tmux send-keys -t $AUTHSESSION "$AUTHSERVER_CMD" C-m
    echo "Ejecutado \"$AUTHSERVER_CMD\" dentro de $AUTHSESSION"
fi

# ============================
# CREAR SESIÓN DE WORLDSERVER
# ============================
if tmux has-session -t $WORLDSESSION 2>/dev/null; then
    echo "La sesión $WORLDSESSION ya está en ejecución."
else
    tmux new-session -d -s $WORLDSESSION
    echo "Sesión de worldserver creada: $WORLDSESSION"
    tmux send-keys -t $WORLDSESSION "$WORLDSERVER_CMD" C-m
    echo "Ejecutado \"$WORLDSERVER_CMD\" dentro de $WORLDSESSION"
fi
```

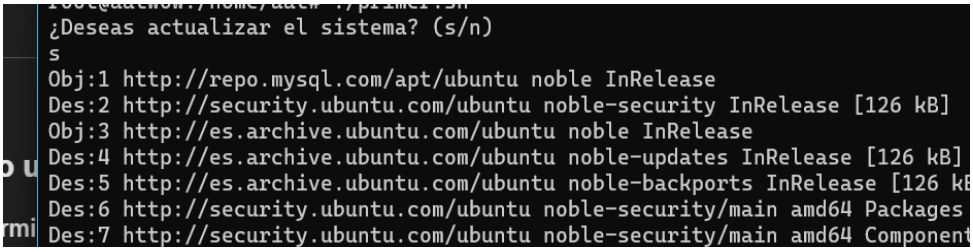
Y por ultimo la información para el usuario:

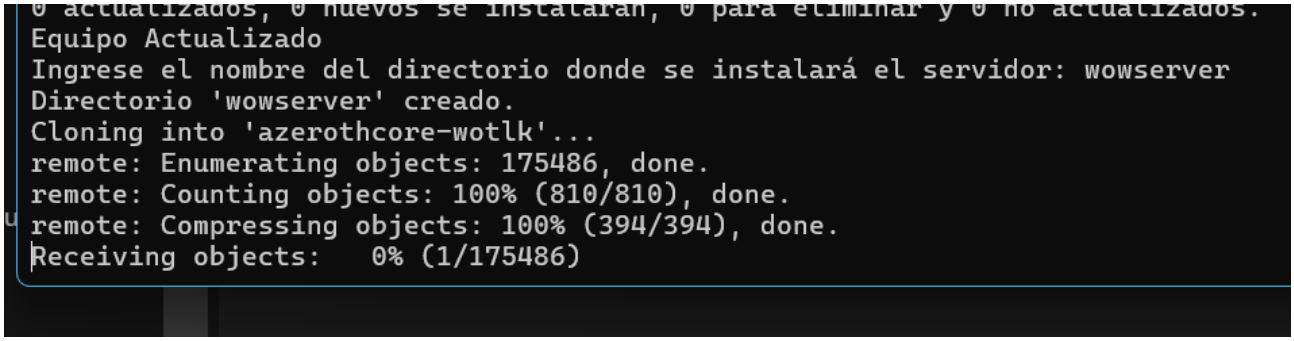
```
# ============================
# INFORMACIÓN PARA EL USUARIO
# ============================
echo ""
echo "💡 Para adjuntarte a una sesión en ejecución usa:"
echo "   tmux attach -t $AUTHSESSION  # Para authserver"
echo "   tmux attach -t $WORLDSESSION  # Para worldserver"
echo "   tmux ls  # Para ver todas las sesiones activas"
```

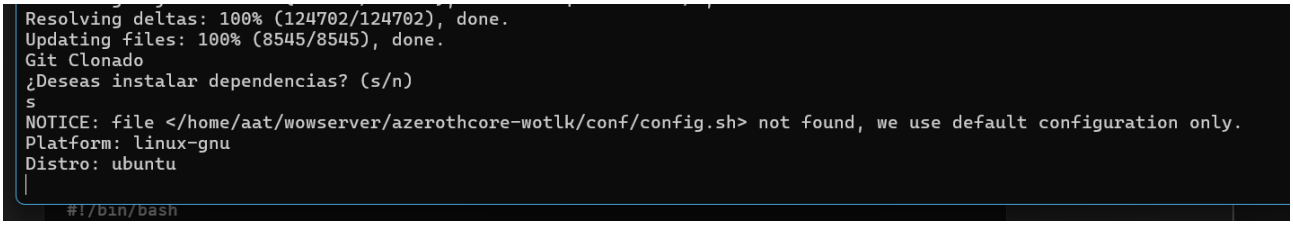
Algunas Comprobaciones principales en Maquina Virtual (no puedo comprobar todo por falta de tiempo):

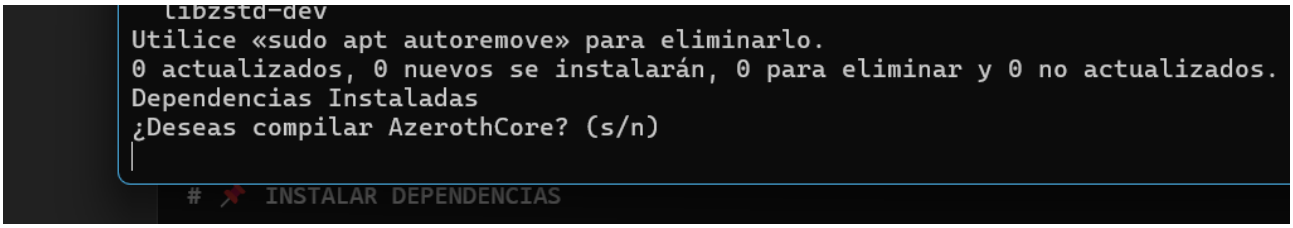
Se les da permisos al los 3 principales:

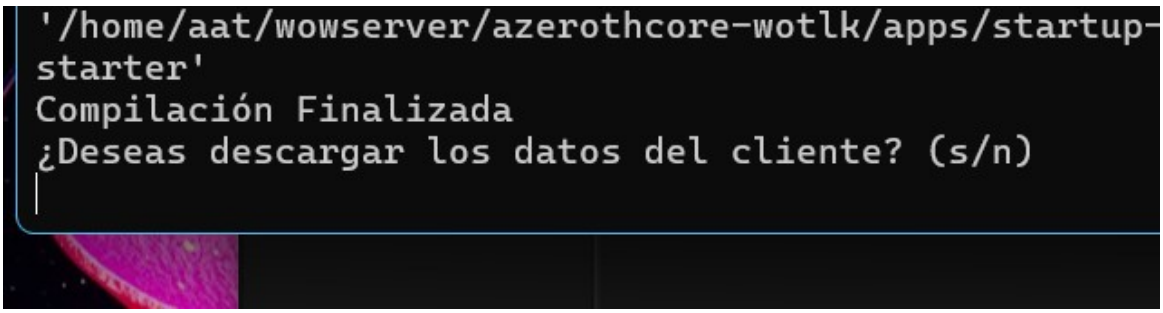
Con chmod +x “script”
Paso a paso:











Vemos que pasa si ponemos un error

```
Creando usuario MySQL...
Ingrese el nombre del usuario a crear: wowaso
Ingrese la contraseña para wowaso:
Error: La contraseña no puede estar vacía.
```

Vemos el warning que no salta en la contraseña mysql:

```
Ingrese la contraseña de MySQL:
mysql: [Warning] Using a password on the command line interface can be insecure.
```

Mejoras extra En logs, se puede añadir en todos los scripts lo siguiente para que guarde los logs en la ruta /var/log/:

```
LOG_FILE="/var/log/instalacion_wow.log"
exec > >(tee -a "$LOG_FILE") 2>&1
```

Asi si nos vamos o lo dejamos ausente mientras instala, podremos recurrir al log si falla algo, ademas en el primer añadiremos la fecha y eliminamos los mensajes de estado, repetiremos esto en los scripts añadiendo la grabacion en el logfile + la fecha:

```
# ================================
# 🚀 INICIO DE INSTALACIÓN
# ================================
echo "========================================"
echo "🚀 Inicio de instalación: $(date)"
echo "========================================"
```