#!/bin/bash

# Function to run a shell command and handle errors

run\_command() {

echo -e "\n$2"

eval $1

if [ $? -ne 0 ]; then

echo "Warning: $2 encountered an error, but continuing with the script."

echo "You may want to check your package repositories for issues."

fi

}

# Function to pause before closing

pause() {

read -p "Press any key to continue or Ctrl+C to exit..."

}

# Step 1: Install Dependencies

install\_dependencies() {

echo "Step 1: Installing dependencies..."

run\_command "sudo apt update" "Updating package lists..."

run\_command "sudo apt upgrade -y" "Upgrading system packages..."

run\_command "sudo apt install cmake gcc g++ git build-essential libpng-dev python3-pip wget curl -y" "Installing build tools and dependencies..."

run\_command "sudo apt install libudev1 libudev-dev -y" "Installing libudev libraries..."

}

# Step 2: Install Box64

setup\_box64() {

echo "Step 2: Setting up Box64..."

run\_command "git clone https://github.com/ptitSeb/box64.git" "Cloning Box64 repository..."

run\_command "cd box64 && mkdir build && cd build && cmake .. -DCMAKE\_BUILD\_TYPE=RelWithDebInfo && make" "Building Box64 from source..."

run\_command "cd box64/build && sudo make install" "Installing Box64..."

run\_command "sudo ldconfig" "Refreshing library paths for Box64..."

}

# Step 3: Install Box86

setup\_box86() {

echo "Step 3: Setting up Box86..."

run\_command "git clone https://github.com/ptitSeb/box86.git" "Cloning Box86 repository..."

run\_command "cd box86 && mkdir build && cd build && cmake .. -DCMAKE\_BUILD\_TYPE=RelWithDebInfo && make" "Building Box86 from source..."

run\_command "cd box86/build && sudo make install" "Installing Box86..."

run\_command "sudo ldconfig" "Refreshing library paths for Box86..."

}

# Step 4: Install Wine

setup\_wine() {

echo "Step 4: Installing Wine..."

run\_command "sudo dpkg --add-architecture i386" "Adding i386 architecture for Wine..."

run\_command "sudo mkdir -pm755 /etc/apt/keyrings" "Creating keyrings directory..."

run\_command "sudo wget -O /etc/apt/keyrings/winehq-archive.key https://dl.winehq.org/wine-builds/winehq.key" "Downloading WineHQ key..."

run\_command "sudo wget -NP /etc/apt/sources.list.d/ https://dl.winehq.org/wine-builds/ubuntu/dists/noble/winehq-noble.sources" "Setting up Wine repository..."

run\_command "sudo apt update" "Updating package list after adding Wine repository..."

run\_command "sudo apt install --install-recommends winehq-stable -y" "Installing Wine Stable version..."

}

# Step 5: Install DXVK

setup\_dxvk() {

echo "Step 5: Installing DXVK..."

run\_command "wget https://github.com/doitsujin/dxvk/releases/download/v2.3/dxvk-2.3.tar.gz" "Downloading DXVK package..."

run\_command "tar -xvf dxvk-2.3.tar.gz" "Extracting DXVK package..."

run\_command "cd dxvk-2.3 && sudo ./setup\_dxvk.sh install" "Installing DXVK..."

}

# Step 6: Install Steam

setup\_steam() {

echo "Step 6: Installing Steam for 64-bit architecture..."

# Remove i386 architecture to avoid compatibility issues

run\_command "sudo dpkg --remove-architecture i386" "Removing i386 architecture..."

run\_command "sudo add-apt-repository multiverse -y" "Enabling multiverse repository..."

run\_command "sudo apt update" "Updating package list after adding multiverse repository..."

# Attempt to install Steam

run\_command "sudo apt install steam -y" "Installing Steam (64-bit only)..."

# If the installation fails, we can try a fallback method

if ! command -v steam > /dev/null; then

echo "Installing Steam using Flatpak as a fallback..."

run\_command "sudo apt install flatpak -y" "Installing Flatpak..."

run\_command "flatpak install flathub com.valvesoftware.Steam -y" "Installing Steam via Flatpak..."

fi

}

# Function to check if Steam is installed

check\_steam\_installed() {

if ! command -v steam > /dev/null; then

echo "Steam is not installed. Please install Steam first using your package manager or Box64."

read -p "Press any key to exit..."

exit 1

fi

}

# Function to run Steam via Box64

run\_steam() {

echo "Running Steam using Box64..."

# Set the environment for Box64, Wine, and DXVK

export BOX86\_PATH=/usr/local/bin/box86

export BOX64\_PATH=/usr/local/bin/box64

export WINEPREFIX=$HOME/.wine

export DXVK\_LOG\_LEVEL=none

export WINEARCH=win64

# Check for the existence of libudev.so.0

UDEV\_LIB\_PATH=$(find /lib/x86\_64-linux-gnu/ -name "libudev.so.0")

if [ -n "$UDEV\_LIB\_PATH" ]; then

export BOX64\_EMULATED\_LIBS="$UDEV\_LIB\_PATH"

echo "BOX64\_EMULATED\_LIBS set to $UDEV\_LIB\_PATH"

else

echo "Error: libudev.so.0 not found. Please install it or check your library paths."

exit 1

fi

# Ensure the necessary libraries for Wine and DXVK are properly linked

sudo ldconfig

# Check for Vulkan driver presence

if ! command -v vulkaninfo > /dev/null; then

echo "Warning: Vulkan drivers may not be installed. Steam might not run properly."

fi

# Launch Steam through Box64

echo "Attempting to launch Steam..."

$BOX64\_PATH /usr/games/steam

if [ $? -ne 0 ]; then

echo "Error: Steam failed to launch. Check for additional errors above."

exit 1

fi

}

# Main function

main() {

echo "Starting installation process..."

pause # Pause before starting the installation

install\_dependencies

setup\_box64

setup\_box86

setup\_wine

setup\_dxvk

setup\_steam

echo "All installations are complete!"

# After installation, run Steam

check\_steam\_installed

run\_steam

# Final pause before closing

read -p "Press any key to exit..."

}

main