Re: Need your help

From: Rolf G (rgiegeri@yahoo.com)

To: tdunn599@yahoo.com

Date: Friday, November 15, 2024 at 09:03 AM EST

Hi Ted,

Modifying the autoexec.sh should meet your needs. I used ChatGPT to help build this script. Replace the entry in autoexec.sh with the code below. This script will require your "dev" python script to be located in the kiosk folder too. You can change the script to use your "dev" file name by updating line: echo "2. kiosk_flip_dev.py"
Full details below. Let me know if any questions or how it goes.

Rolf

```
#!/bin/bash

cd kiosk

function choose_script() {
  while true; do
    echo "Select a script to run:"
    echo "1. kiosk_flip.py"
    echo "2. kiosk_flip_dev.py"
    read -r -t 5 -p "Enter your choice (1 or 2): " choice

    # Check if user input is empty or not a number
    if [[ -z "$choice" || ! "$choice" =~ ^[1-2]$ ]]; then
      if [[ -z "$choice" ]]; then
        echo "No input received. Defaulting to 1."
        choice=1
        break
      else
        echo "Invalid input. Please enter 1 or 2."
        continue
      fi
    fi

    break
  done

  case $choice in
    1)
      echo "Running kiosk_flip.py..."
      python kiosk_flip.py
      ;;
    2)
      echo "Running kiosk_flip_dev.py..."
      python kiosk_flip_dev.py
      ;;
  esac
}

choose_script
```

What Happens When the Script is Run:
1. The script starts and changes the working directory to `kiosk`.
2. The `choose_script` function is invoked.
3. The user is shown the following prompt with a 5-second timeout:
   ```
   Select a script to run:
   1. kiosk_flip.py
   2. kiosk_flip_dev.py
   Enter your choice (1 or 2):
   ```
4. If the user provides `1` or `2`, the corresponding Python script is executed.
5. If no input is given within 5 seconds, the script defaults to running `kiosk_flip.py`.
6. If the user provides an invalid input (other than `1` or `2`), they are asked to enter a valid choice again.

Detailed Explanation:

1. **Shebang Line (`#!/bin/bash`)**

```bash
#!/bin/bash
```
- This is the "shebang" line, and it tells the operating system that the script should be run using the Bash shell (`/bin/bash`).
- This is necessary for scripts to run in a specific shell environment, ensuring compatibility.

2. **Change Directory (`cd kiosk`)**

```bash
cd kiosk
```
- This line changes the current working directory to `kiosk`. All subsequent commands will be executed inside the `kiosk` directory.
- The script assumes that the directory `kiosk` exists, and it contains the two Python scripts (`kiosk_flip.py` and `kiosk_flip_dev.py`) that the user will choose from.

3. **Function Definition (`choose_script`)**

```bash
function choose_script() {
```
- This line defines a function named `choose_script`. All the logic to prompt the user, validate their input, and run the selected script is contained within this function.
- Functions allow you to group and organize code logically, and the function will be invoked later in the script.

4. **Infinite Loop (`while true`)**

```bash
while true; do
```
- This starts an infinite loop (`while true`) that will repeatedly ask the user for input until they provide a valid response.
- The loop ensures that invalid input is handled and re-prompts the user until they provide acceptable input or timeout occurs.

5. **Prompt the User for Input (`read`)**

```bash
echo "Select a script to run:"
echo "1. kiosk_flip.py"
echo "2. kiosk_flip_dev.py"
read -r -t 5 -p "Enter your choice (1 or 2): " choice
```
- The `echo` commands display the menu options to the user:
  - `1. kiosk_flip.py`
  - `2. kiosk_flip_dev.py`

- The `read` command prompts the user for input. Specifically:
  - `-r`: This option prevents backslashes from escaping characters (useful for reading raw input).
  - `-t 5`: This sets a timeout of 5 seconds. If the user does not provide input within this time, the `read` command will exit and the `$choice` variable will remain empty.
  - `-p "Enter your choice (1 or 2): "`: This prints the prompt message to the terminal, asking the user to enter their choice.
  - The result is stored in the variable `choice`.

6. **Check for Empty or Invalid Input**

```bash
if [[ -z "$choice" || ! "$choice" =~ ^[1-2]$ ]]; then
  if [[ -z "$choice" ]]; then
    echo "No input received. Defaulting to 1."
    choice=1
    break
  else
    echo "Invalid input. Please enter 1 or 2."
    continue
  fi
fi
```

- **Empty Input (`-z "$choice"`)**:
  - If the user doesn't enter anything within 5 seconds (because of the `-t 5` timeout), the `$choice` variable will be empty, and the condition `-z "$choice"` will evaluate to `true`.

  - If this happens, the script will print `"No input received. Defaulting to 1."` and set `$choice=1` as the default option. The loop is then exited using the `break` statement.

- **Invalid Input (`! "$choice" =~ ^[1-2]$`)**:
  - If the input is neither `1` nor `2`, the regular expression (`^[1-2]$`) will fail, and the condition will evaluate to `true`.

  - In this case, the script will print `"Invalid input. Please enter 1 or 2."`, and the `continue` command will be executed, which causes the loop to start over and ask the user for input again.

7. **Break the Loop**

```bash
break
```
- This `break` statement is executed if the user provides a valid input or if the input is empty (after 5 seconds), in which case the script defaults to `1`.
- After `break` is executed, the loop exits, and the script proceeds to the next part: the `case` statement.

8. **Select the Script to Run Using a `case` Statement**

```bash
case $choice in
  1)
    echo "Running kiosk_flip.py..."
    python kiosk_flip.py
    ;;
  2)
    echo "Running kiosk_flip_dev.py..."
    python kiosk_flip_dev.py
    ;;
esac
```
- This `case` statement checks the value of the `$choice` variable.
  - If `$choice` is `1`, the script will print `"Running kiosk_flip.py..."` and then run the `kiosk_flip.py` Python script using the `python` command.
  - If `$choice` is `2`, the script will print `"Running kiosk_flip_dev.py..."` and then run the `kiosk_flip_dev.py` Python script.

- The `;;` indicates the end of each `case` block.

9. **Invoke the Function**

```bash
choose_script
```
- Finally, the `choose_script` function is called to start the process.

> On Thursday, November 14, 2024 at 05:49:25 PM EST, Rolf G <rgiegeri@yahoo.com> wrote:
>
> Hi Ted,
>
> I think I can assist. Paste a copy of the
> python kiosk_flip.py and the development scripts in the email and I'll see what I can do.
>
> Please paste the text from the scripts, not attach the.sh file, as the email filter will block it.
>
> Thanks
>
> Rolf
>
> > On Thu, Nov 14, 2024 at 3:30 PM, tdunn599
> > <tdunn599@yahoo.com> wrote:
> >
> > Hi Rolf,
> >
> > I hope everything is fine with you. Matthew told me you are in Canada and suggested I contact you for some help in Linux.
> >
> > Here is my problem:
> >
> > I have the kiosk programs that runs in python and I am using pygame, set to full screen mode, for the

graphics.

There is no way, that I have found, to exit the full screen mode in pygame.

So, I have one program that runs in full screen mode for normal operations and a second program that is identical except it is not full screen mode, for development.

To get to the second non-full screen mode program to run, I have to reboot the system and as the python script is running, hit the Ctrl-C keys to stop the python script. This is OK for Raspberry Pi 0 through 3s, but the Pi-5s are so fast that I sometimes miss.

So, there are two solutions that I see:

1- Find a way to exit full screen mode in pygame.

2- Create a launch program in Linux or python that allows a user input to switch to the alternate version, but times-out after say 5 seconds if the user does not enter anything and then allows the full screen version to run.

I am using a Linux shell script to launch the python program on startup.

Here is a link that describes that system:

https://dunn-itvideoservicesllc.yolasite.com/resources/RaspberryPi/Run_Program_on_startup

The script is ***autoexec.sh.***

It has two lines in it:

cd kiosk

python kiosk_flip.py

Any thought you have on a solution will be greatly appreciated.

Regards, Ted