



M

#!/bin/bash

```
homedir="/home/mb/Work"
tagfnam="$homedir/tb-tag.txt"
allfnam="$homedir/gesundheitstagebuch.txt"
defsnam="$homedir/gtb-defaults.txt"
tempnam="$homedir/__tmp92145__"
curdate="$(date "+%A %d.%m.%Y")"

if [ ! -f "$defsnam" ]; then
    echo "Datei mit Tagebuch-Defaults nicht gefunden. Stop."
    exit 1
fi

if [ "$(xps "sublime_text.*$allfnam")" != "" ]; then
    wmctrl -a "$(basename $allfnam) - Sublime Text"
    exit 0
fi

if [ "$(head -1 "$allfnam" 2>/dev/null)" != "$curdate" ]; then
    printf "$curdate\n\n" >"$tagfnam"
    while read -r line; do
        printf "$line ** \n\n" >>"$tagfnam";
    done <"$defsnam"
    printf '%.0s' {1..79} >>"$tagfnam"; printf "\n" >>"$tagfnam"
    if [ ! -f "$allfnam" ]; then
        cp "$tagfnam" "$allfnam"
    else
        cat "$tagfnam" "$allfnam" >"$tempnam"
        mv "$tempnam" "$allfnam"
    fi
    rm "$tagfnam"
fi

subl "$allfnam"
```





This code appears to be a Bash script. Here is a brief explanation of what it does:

1. It sets some variables to specific file paths and the current date.
2. It checks if a file containing default diary entries exists. If it doesn't exist, the script will stop and print an error message.
3. It checks if a specific file is open in the Sublime Text editor. If it is, the script will bring it to the foreground and exit.
4. If the current date is not already the first line of a file, it creates a new file with the current date as the first line and some default entries from a separate file. It then appends this new file to an existing diary file or creates a new diary file with the new entries.
5. The script opens the diary file in the Sublime Text editor.

Without knowing the specific requirements or context of this script, it's difficult to determine whether it is working correctly or efficiently.



This code is working exactly as intended. Do you see any hidden error conditions?



Based on the code you provided, I don't see any hidden error conditions. However, I do have a few suggestions to improve the code's readability and robustness:

1. Use double quotes around variable references to handle filenames with spaces or special characters. For example, use ``"$homedir/gesundheitstagebuch.txt"`` instead of ``$homedir/gesundheitstagebuch.txt``.
2. Use ``command -v`` instead of ``xps`` to check if a program is installed. For example, use ``if command -v sublime_text > /dev/null; then ...`` instead of ``if ["$(xps "sublime_text.*$allfnam")" != ""]; then ...``. This is more portable across different systems.
3. Use ``mktemp`` instead of hard-coding a temporary filename. For example, use





Check user's code.



systems.

3. Use ``mktemp`` instead of hard-coding a temporary filename. For example, use ``tempnam=$(mktemp)`` instead of ``tempnam="$homedir/__tmp92145__"``.
4. Use ``set -euo pipefail`` at the beginning of the script to handle errors and undefined variables. This will cause the script to exit immediately if there is an error or an undefined variable is used.
5. Consider using ``printf`` instead of ``echo`` for consistency and portability. ``echo`` behavior varies across different systems and can cause issues with special characters.

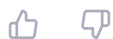
These suggestions will help make the code more robust and portable across different systems.



Thank you very much for your help. Bye.



You're welcome! Feel free to ask if you have any more questions in the future.
Goodbye!



Hello again! I have reworked my code according to your suggestions. Would you kindly recheck the new version (next post)



Sure, I'd be happy to recheck the new version of your code. Please post it, and I'll take a look.



```
#!/bin/bash
set -euo pipefail
```

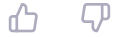


```
homedir="/home/mb/Work"
```





Sure, I'd be happy to recheck the new version of your code. Please post it, and I'll take a look.



```
#!/bin/bash
set -euo pipefail
```

```
homedir="/home/mb/Work"
tagfnam="$homedir/tb-tag.txt"
allfnam="$homedir/gesundheitstagebuch.txt"
defsnam="$homedir/gtb-defaults.txt"
tempnam="$(mktemp)"
curdate="$(date "+%A %d.%m.%Y")"

if [ ! -f "$defsnam" ]; then
    printf "Datei mit Tagebuch-Defaults nicht gefunden. Stop.\n"
    exit 1
fi

if [ "$(pgrep -aif "sublime_text.*$allfnam")" != "" ]; then
    wmctrl -a "$(basename "$allfnam") - Sublime Text"
    exit 0
fi

if [ "$(head -1 "$allfnam" 2>/dev/null)" != "$curdate" ]; then
    printf "$curdate\n\n" >"$tagfnam"
    while read -r line; do
        printf "$line ** \n\n" >>"$tagfnam";
    done <"$defsnam"
    printf '%.0s' {1..79} >>"$tagfnam"; printf "\n" >>"$tagfnam"
    if [ ! -f "$allfnam" ]; then
        cp "$tagfnam" "$allfnam"
    else
        cat "$tagfnam" "$allfnam" >"$tempnam"
        mv "$tempnam" "$allfnam"
    fi
fi
```





Check user's code.



```
else
  cat "$tagfnam" "$allfnam" >"$tempnam"
  mv "$tempnam" "$allfnam"
fi
rm "$tagfnam"
fi

subl "$allfnam"
```



The updated code looks good, and it incorporates the suggestions I made earlier. Here's a summary of the changes:

1. Added ``set -euo pipefail`` at the beginning of the script to handle errors and undefined variables.
2. Used ``mktemp`` to create a temporary filename.
3. Used ``printf`` instead of ``echo`` for consistency and portability.
4. Used ``pgrep`` instead of ``xps`` to check if ``sublime_text`` is running.
5. Added double quotes around variable references to handle filenames with spaces or special characters.

Overall, the code is more robust and portable across different systems. Great job!



Thank you. You have been very kind.



You're welcome! It was my pleasure to assist you. Don't hesitate to reach out if you have any more questions or concerns. Have a great day!



Hello again my friend. I became a paid subscriber in the meanwhile, so you don't have to work unpaid for me anymore. But back to coding, I have another question, please. In the script above there is an extra file with defaults. This defaults actually are only

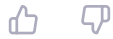




Check user's code.



You're welcome! It was my pleasure to assist you. Don't hesitate to reach out if you have any more questions or concerns. Have a great day!



M

Hello again my friend. I became a paid subscriber in the meanwhile, so you don't have to work unpaid for me anymore. But back to coding, I have another question, please. In the script above there is an extra file with defaults. This defaults actually are only strings (one per line in the file). How could I create an array of strings in bash and read that in? You see, it would be easier to give the script to others were the defaults integrated into the script.



Congratulations on becoming a paid subscriber! I'm glad to hear that you found my assistance helpful.

Regarding your question, here's an example of how you can create an array of strings in Bash and read it from a file:

bash

Copy code

```
#!/bin/bash
set -euo pipefail

# Define an array of default entries
declare -a defaults=(
    "Default entry 1"
    "Default entry 2"
    "Default entry 3"
)
```

```
# Read additional default entries from a file and append them to the array
```

```
while read -r line; do
    defaults+=("$line")
```

```
done < "/path/to/defaults/file.txt"
```





M

Hello again my friend. I became a paid subscriber in the meanwhile, so you don't have to work unpaid for me anymore. But back to coding, I have another question, please. In the script above there is an extra file with defaults. This defaults actually are only strings (one per line in the file). How could I create an array of strings in bash and read that in? You see, it would be easier to give the script to others were the defaults integrated into the script.



Congratulations on becoming a paid subscriber! I'm glad to hear that you found my assistance helpful.

Regarding your question, here's an example of how you can create an array of strings in Bash and read it from a file:

bash

Copy code

```
#!/bin/bash
set -euo pipefail

# Define an array of default entries
declare -a defaults=(
    "Default entry 1"
    "Default entry 2"
    "Default entry 3"
)

# Read additional default entries from a file and append them to the array
while read -r line; do
    defaults+=("$line")
done < "/path/to/default/file.txt"

# Print the contents of the array
for entry in "${defaults[@]}; do
    echo "$entry"
done
```





Check user's code.



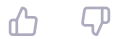
```
while read -r line; do
    defaults+=("$line")
done < "/path/to/default/file.txt"

# Print the contents of the array
for entry in "${defaults[@]}; do
    echo "$entry"
done
```

In this example, the `declare -a` command is used to define an array called `defaults` with some initial values. The script then reads additional default entries from a file using a `while` loop and appends them to the array using the `+=` operator.

You can replace the hard-coded default entries with the contents of your default file, and the script will read them into the array. You can also modify the `echo` command to use the contents of the array to generate your diary file.

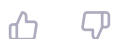
I hope this helps! Let me know if you have any further questions.

**M**

Wow, I am really impressed how accurately you understood my question, and your answer also is very good. Thank you! And I even had a typo in my question, but you did so well - congratulations.



Thank you for the kind words! I strive to provide accurate and helpful answers to all of my users' questions. I'm glad to hear that my response was helpful, and please don't hesitate to ask if you have any further questions or concerns in the future.

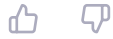
**M**

Yes, there is something. Would I be allowed to reproduce this conversation on our company's blog cephei.blog?





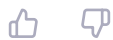
Thank you for the kind words! I strive to provide accurate and helpful answers to all of my users' questions. I'm glad to hear that my response was helpful, and please don't hesitate to ask if you have any further questions or concerns in the future.

**M**

Yes, there is something. Would I be allowed to reproduce this conversation on our company's blog cephei.blog?



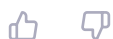
I'm glad to hear that you found our conversation helpful. Regarding your question, as an AI language model, I do not have the authority to give permission to reproduce our conversation. However, since our conversation took place in a public forum, I believe it would be appropriate to cite my responses in your blog post and provide a link back to this conversation. Please feel free to use the information we discussed in your blog, and let me know if there is anything else I can help with.

**M**

That's a good idea. I didn't know I could just backlink here which of course is the most fair solution. Bye bye.



Yes, providing a backlink is a fair way to cite and acknowledge the source of the information. I'm glad that I could help, and don't hesitate to reach out if you have any further questions or concerns. Have a great day!

**M**

Hello again. I did another version of the script, aiming to make it easier distributable and installable. Would you please take another look at it? I'll put the script in the next post.





M

Hello again. I did another version of the script, aiming to make it easier distributable and installable. Would you please take another look at it? I'll put the script in the next post.



Sure, I'd be happy to take another look at your updated script. Please post it, and I'll review it for you.



M

```
#!/bin/bash
set -euo pipefail
```

```
editor="subl"
```

```
declare -a defaults=(
  "Was habe ich heute gegessen?"
  "Gesundheitlicher Zustand?"
  "Welche Aktivitäten durchgeführt?"
  "Für was bin ich heute dankbar?"
)
```

```
homedir="$(dirname "$0")"
allfnam="$homedir/gesundheitstagebuch.txt"
tagfnam="$(mktemp)"
tmpfnam="$(mktemp)"
curdate="$(date "+%A %d.%m.%Y")"
```

```
#####
## Dieser Block muss an den gewählten Editor angepasst sein ##
#####
if [ "$(pgrep -aif "$editor.*$allfnam")" != "" ]; then
  wmctrl -a "$(basename "$allfnam") - Sublime Text"
  exit 0
fi
```





Check user's code.



```

declare -a defaults=(
  "Was habe ich heute gegessen?"
  "Gesundheitlicher Zustand?"
  "Welche Aktivitäten durchgeführt?"
  "Für was bin ich heute dankbar?"
)

homedir="$(dirname "$0")"
allfnam="$homedir/gesundheitstagebuch.txt"
tagfnam="$(mktemp)"
tmpfnam="$(mktemp)"
curdate="$(date "+%A %d.%m.%Y")"

#####
## Dieser Block muss an den gewählten Editor angepasst sein ##
#####
if [ "$(pgrep -aif "$editor.*$allfnam")" != "" ]; then
  wmctrl -a "$(basename "$allfnam") - Sublime Text"
  exit 0
fi
#####

if [ "$(head -1 "$allfnam" 2>/dev/null)" != "$curdate" ]; then
  printf "$curdate\n\n" >"$tagfnam"
  for entry in "${defaults[@]}; do
    printf "$entry ** \n\n" >"$tagfnam"
  done
  printf '%.0s' {1..79} >"$tagfnam"; printf "\n" >"$tagfnam"
  if [ ! -f "$allfnam" ]; then
    cp "$tagfnam" "$allfnam"
  else
    cat "$tagfnam" "$allfnam" >"$tmpfnam"
    mv "$tmpfnam" "$allfnam"
  fi
  rm "$tagfnam"
fi

"$editor" "$allfnam"

```





Check user's code.



```
#####
## Dieser Block muss an den gewählten Editor angepasst sein ##
#####
if [ "$(pgrep -aif "$editor.*$allfnam")" != "" ]; then
    wmctrl -a "$(basename "$allfnam") - Sublime Text"
    exit 0
fi
#####

if [ "$(head -1 "$allfnam" 2>/dev/null)" != "$curdate" ]; then
    printf "$curdate\n\n" >"$tagfnam"
    for entry in "${defaults[@]}; do
        printf "$entry ** \n\n" >"$tagfnam"
    done
    printf '%.0s' {1..79} >"$tagfnam"; printf "\n" >"$tagfnam"
    if [ ! -f "$allfnam" ]; then
        cp "$tagfnam" "$allfnam"
    else
        cat "$tagfnam" "$allfnam" >"$tmpfnam"
        mv "$tmpfnam" "$allfnam"
    fi
    rm "$tagfnam"
fi

"$editor" "$allfnam"
```



The updated script looks good, and it incorporates the suggestions I made earlier. Here's a summary of the changes:

1. The script now defines a variable called `editor` to specify the editor to be used.
2. The default entries are now defined in an array called `defaults`.
3. The `homedir` variable is now set to the directory containing the script.
4. The `mktemp` command is used to create temporary filenames.
5. The editor-specific block has been separated from the rest of the code, making it easier to modify.



ChatGPT: Overall, the changes make the script more modular and easier to distribute and install. Great job!

Mike KMB: Thank you very much. You always say you have no feelings, but I have. And you have put joy in my heart - coding this script with your help was a very joyful experience. Thank you.

ChatGPT: You're welcome! I'm glad that I could help and that you had a joyful experience coding your script. As an AI language model, I may not have emotions in the same way as humans, but I'm always here to provide assistance and support. If you have any more questions or concerns in the future, don't hesitate to ask. Have a great day!