Instantly share code, notes, and snippets.

nathants / **disable_keyboard.sh**

Last active 7 months ago

Star

Code   Revisions 3

after some work minutes, lock the screen requiring password entry, and disable the keyboard for a few minutes, delaying possibility of unlock and forcing break minutes

disable_keyboard.sh

```bash
#!/bin/bash

# The MIT License (MIT)
# Copyright (c) 2022-present Nathan Todd-Stone
# https://en.wikipedia.org/wiki/MIT_License#License_terms

set -euo pipefail

seconds=$1

start=$(date +%s)

while true; do
    for id in $(xinput list|grep -e USB -e 'AT Translated Set 2 keyboard' -e 'IBM TrackPoint'|grep -
        xinput disable $id
    done
    now=$(date +%s)
    elapsed=$(($now - $start))
    if (($elapsed > $seconds)); then
        break
    fi
    sleep .25
done

for id in $(xinput list|grep -e USB -e 'AT Translated Set 2 keyboard' -e 'IBM TrackPoint'|grep -Po '
    xinput enable $id
done

sleep 1

xmodmap ~/.xmodmap
```

pomodoro.py

```python
#!/usr/bin/env python3

# The MIT License (MIT)
# Copyright (c) 2022-present Nathan Todd-Stone
# https://en.wikipedia.org/wiki/MIT_License#License_terms

import argh # pip install argh
import blessings # pip install blessings
import queue
import subprocess
import sys
import termios
import threading
import time
import tty

def getch():
    fd = sys.stdin.fileno()
    old = termios.tcgetattr(fd)
    try:
        tty.setraw(fd)
        val = sys.stdin.read(1).lower()
        if val == '\x03':
            sys.exit(1)
        else:
            return val
    except KeyboardInterrupt:
        sys.exit(1)
    finally:
        termios.tcsetattr(fd, termios.TCSADRAIN, old)

def run_thread(fn, *a, **kw):
    obj = threading.Thread(target=fn, args=a, kwargs=kw)
    obj.daemon = True
    obj.start()
    return obj

def enqueue_keypress(q):
    while True:
        q.put(getch()[0])

def _main(q, term, temp_work_minutes, work_minutes, break_seconds):
    last = time.time()
    last_notify = 0
    while True:
        _work_minutes = temp_work_minutes if temp_work_minutes else work_minutes
        try:
            char = q.get(False)
        except queue.Empty:
            pass
        else:
            if char == 'r':
                print(f'{term.clear}{term.move(0,0)}reset work time')
                time.sleep(1)
                last = time.time()
            elif char == 's':
                last = -_work_minutes * 60
        minutes_until_break = (last + _work_minutes * 60 - time.time()) / 60
        if minutes_until_break <= 0:
            if subprocess.check_output('ps -ef | grep -e ffmpeg -e vlc -e zoom | grep -v grep || tru
                msg = 'an important app is open, not locking screen'
                if time.time() - last_notify > 180 and not subprocess.check_output(f'ps -ef | grep n
                    print(f'{term.clear}{term.move(0,0)}{msg}')
                    subprocess.check_output(f'notify {msg} &>/dev/null </dev/null &', shell=True)
                    last_notify = time.time()
            else:
                if temp_work_minutes:
                    temp_work_minutes = 0
                start = time.time()
                subprocess.check_output(f'disable_keyboard.sh {break_seconds} &>/dev/null </dev/null
                subprocess.check_output('sleep 1; slock', shell=True)
                subprocess.check_output(f'notify break was {int((time.time() - start) / 60)} minutes
                last = time.time()
        else:
            print(f'{term.clear}{term.move(0,0)}{"temporary " if temp_work_minutes else ""}work minu
        time.sleep(.1)

def main(temp_work_minutes=0, work_minutes=35, break_seconds=60 * 10):
    term = blessings.Terminal()
    q = queue.Queue(1)
    run_thread(enqueue_keypress, q)
    try:
        with term.hidden_cursor():
            _main(q, term, temp_work_minutes, work_minutes, break_seconds)
    except KeyboardInterrupt:
        print(term.clear)

if __name__ == '__main__':
    argh.dispatch_command(main)
```