Table of contents

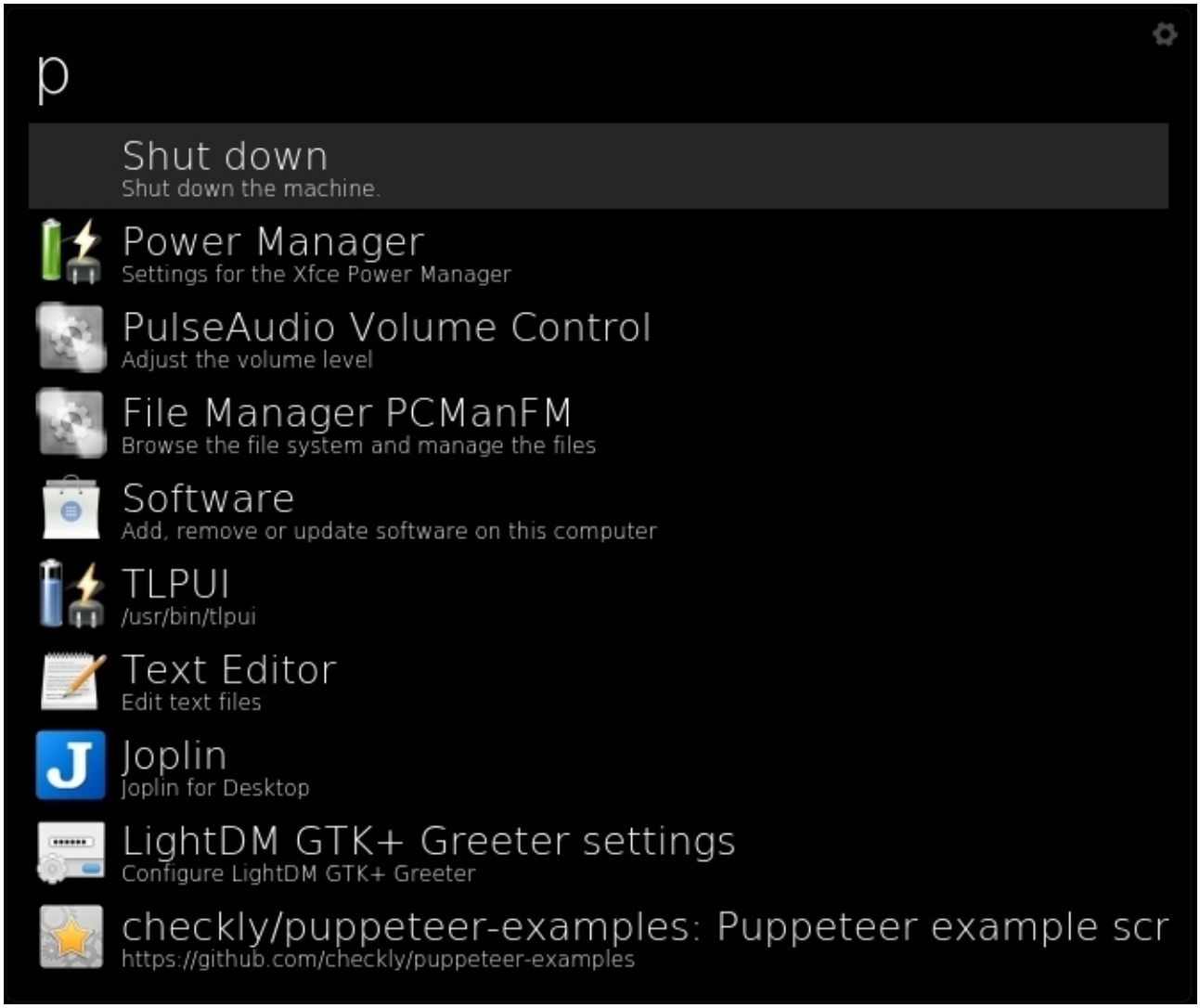
1 내가 리눅스를 써야만 하는 이유 4 - 내겐 너무나 유용한 어플들

1.1 네번째 이유

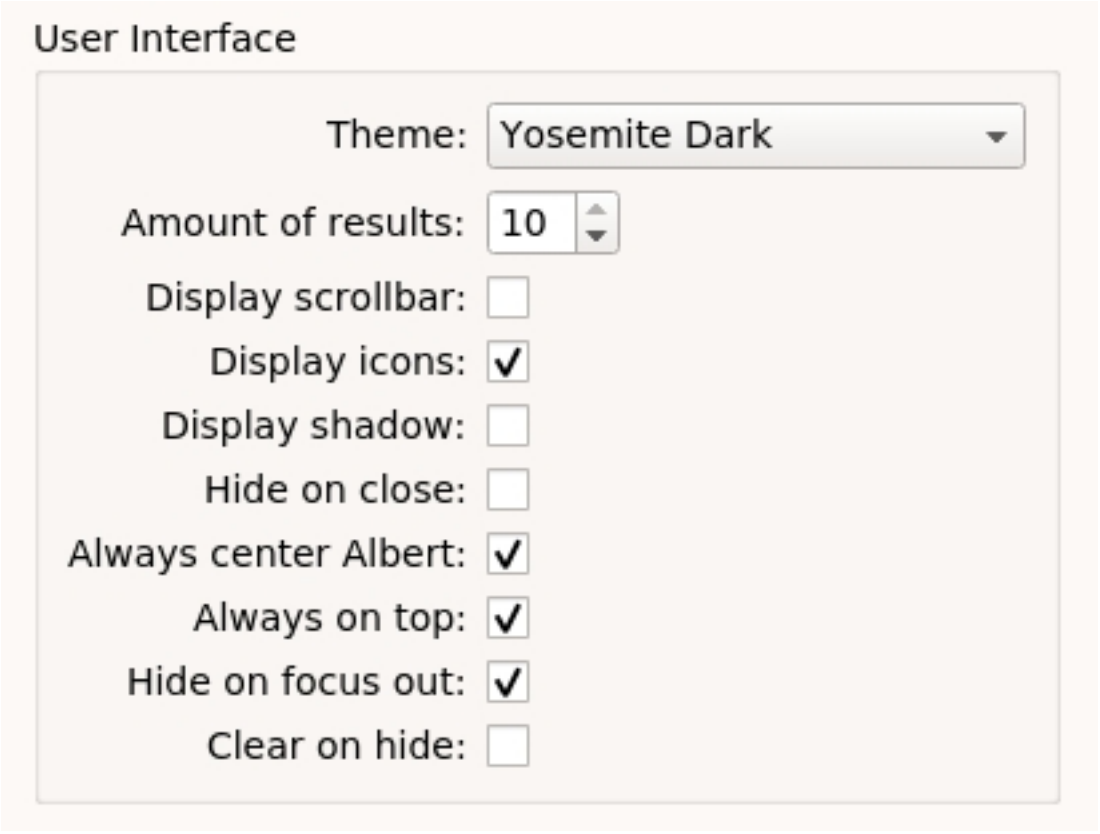
Ubuntu 12.04에 이어 Ubuntu 16.04, 그리고 Lubuntu 18.04까지, 벌써 6년째 리눅스로 매일 업무를 하고, 개인적으로도 1년 넘게 리눅스 랩탑을 사용하다 보니, 하나 둘씩 손에 익은 어플들이 늘어갔다. 물론 이것 저것 테스트해보고, 다른 것 찾아보고 하는 등의 즐거움(= 맨 땅에 헤딩 = 시행착오)도 만만치 않았다. 하지만 쏟아부은 노력과 시간에 비하면 이미 얻어 낸 효용이나 앞으로 얻을 미래의 효용은 훨씬 클 것이라 생각된다. 만약 나중에 업무용으로 윈도우를 사용해야만 하는 상황에 처한다면, 한 동료가 이미 했던 것처럼 VM으로 리눅스를 띄워서 사용할 것 같을 정도이다. 아니면 WSL2를 쓰거나. 하여간, 이렇게나 편리한 환경을 벗어나서 다른 환경으로 옮겨 가기는 쉽지 않아 보인다.

이미 i3wm, docker, 그리고 tmux만 해도 엄청난 생산성 향상이 있지만, 그와 더불어 아래에 소개된, 개인적으로 매일 매일 사용하는 소소한 어플들을 적재적소에 사용할 수 있는 점은, 내가 리눅스를 반드시 써야만 하는 그 이유 중 하나이다.

1.2 albert

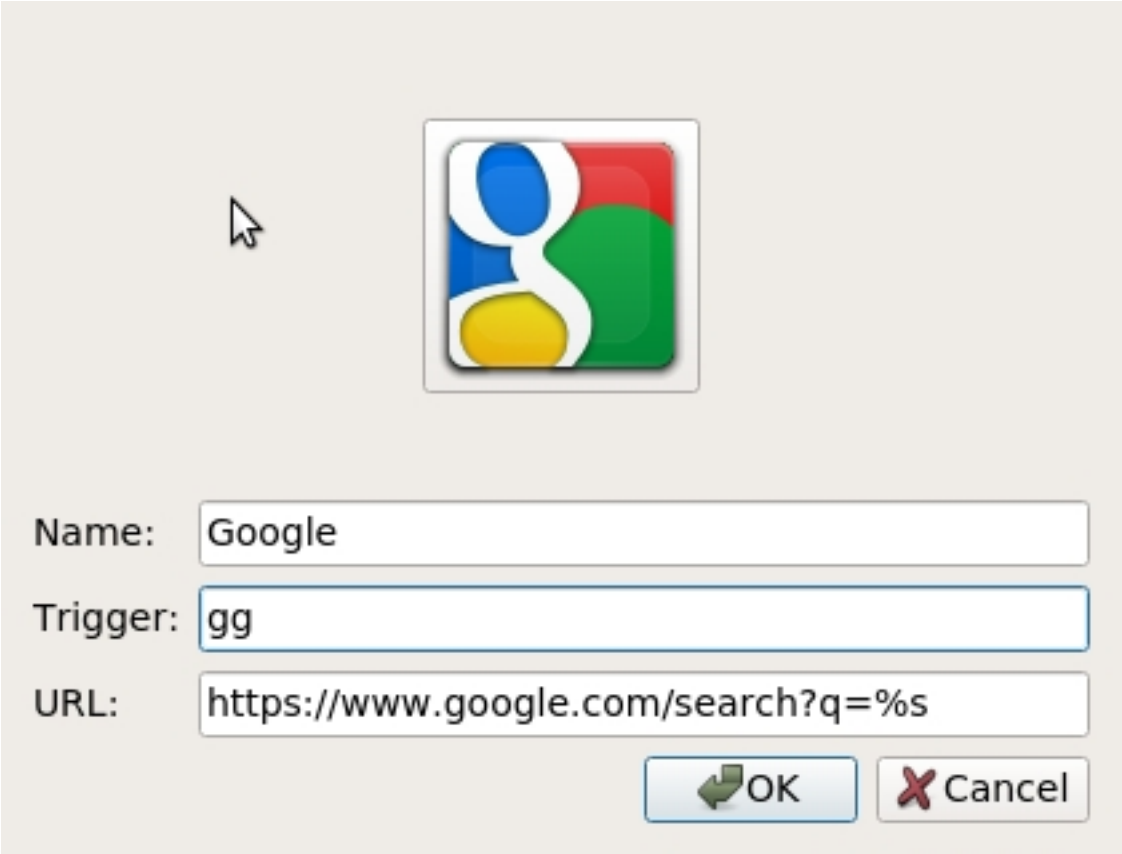


Alfred라고 하는 OSX용 런처 어플의 리눅스용 대체 어플이다. 지정된 Hot Key를 누르면 런쳐 창이 나오고, 타이핑을 하면 선택 가능한 목록이 제공되고, 엔터를 치면 선택한 항목이 실행된다. 개인적으로, 일반 키보드에서는 Meta(= Windows key) + a로, 맥북 키보드에서는 Command를 Option키와 바꾼 후, Option + a로 Hot Key를 지정해서 사용중이다. i3wm과 함께 사용중이라서, Yosemite Dark theme과 함께 Display shadow를 선택하지 않음으로써 그냥 깔끔하게 검정색 박스로 설정했다.



1.2.1 제공되는 기능 중 사용하는 기능들

- Applicatoins: 어플리케이션을 검색해서 실행할 수 있다. 대충 기억나는 글자만 쳐도 원하는 어플을 쉽게 찾을 수 있다.
- Chrome bookmarks: Chrome에 등록된 북마크들을 검색 후 바로 띄울 수 있다. 자주 가는 업무 사이트나 url 들을 북마크에 추가한 후 그냥 기억나는 단어만 타이핑하면 금새 검색할 수 있어서 매우 편리하다.
- Web search: 회사 업무 사이트 내의 검색이나, 구글 드라이브 검색 등, 원하는 검색 url과 `gg` 같은 trigger(공백이 맨 뒤에 있어야 함에 주의)을 등록 후 바로 검색할 수 있다. 예를 들어, 기본적으로 등록된 구글 검색의 경우 trigger가 `gg` 이므로, `gg 불매운동` 처럼 입력 후 엔터를 치면 구글 검색 페이지가 실행중인 크롬에 새 탭으로 열리게 된다. Trigger는 다른 항목에서 사용되지 않은 것이어야 하며, 알파벳의 조합과 공백 하나를 맨 뒤에 두어서 여타의 검색과 구분이 가능야 한다.
- Python: Python extension들이 많은데, 개인적으로 쓰고 있는 건 Window switcher뿐이다. 실행중인 어플들을 검색시 목록에 표시해서 실행중인 어플들로 바로 돌아갈 수 있게 해준다.
- System: 주로 Shutdown만 설정해서 사용중인데, logout, restart 등 시스템 제어 명령을 제어할 수 있다.
- Calculator: `100*2` 같은 수식을 입력하면 바로 결과를 볼 수 있다.
- MPRIS control: MPRIS가 지원되는 VLC같은 미디어 플레이어를 `play`나 `stop` 같은 명령어를 입력해서 제어할 수 있다.



i3wm과 더불어, 남들에게 뭔가를 보여줄 때 약간 뿌듯함을 느끼게 해주는 어플 중 하나이다. 향후 필요한 Python extension을 개발을 위해 Python 공부도 최근에 시작했다.

1.2.2 i3wm에 설정

부팅 후 자동으로 albert를 시작하게 하고, 이후 meta + a를 누르면 albert launcher가 나오게 하려고 한다. vi ~/.i3/config 해서 다음을 추가한 후 재부팅하면 된다.

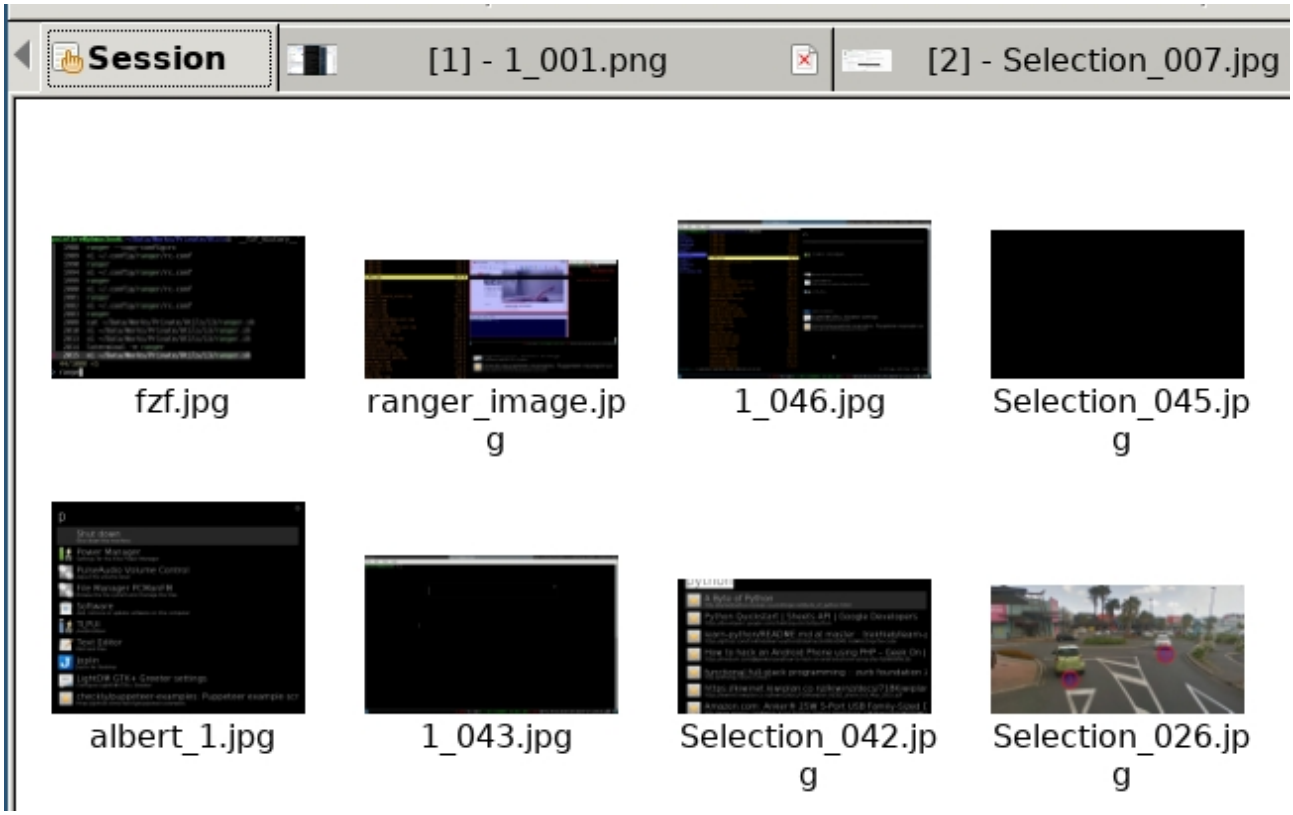
```
##############################
# Albert                     #
##############################
# Pop up Albert window
bindsym $mod+a exec --no-startup-id albert show

# Autostart albert
exec --no-startup-id albert &
```

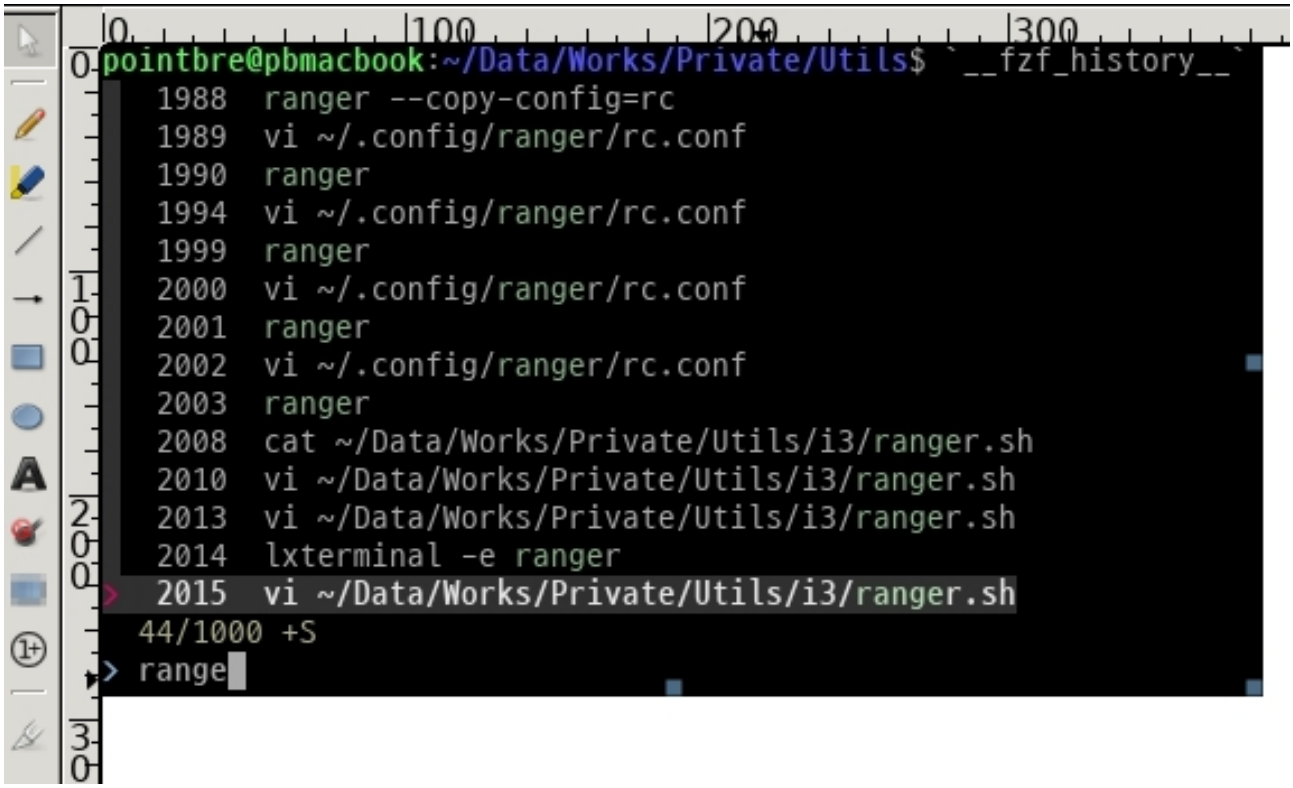
1.3 Shutter

6년전 처음 리눅스를 사용하기 시작했을 때 부터 지금까지 유용하게 쓰는 스크린샷 어플리케이션이다. 요즘 인기가 있는 Flameshot이나 Ksnip도 잠깐 써 봤지만, 이 오래된 어플리케이션이 제공하는 편리함을 따라잡지 못하는 거 같다.

먼저 다양한 캡쳐 방법을 들 수 있다. 영역/윈도우/윈도우 내 서브 윈도우/전체/메뉴/웹/툴팁과 함께 delay 옵션을 제공해준다. 메뉴 캡쳐 기능은 몇 번 정말 유용하게 사용했다.

과거 스크린샷들을 볼 수 있는 세션과 자체 편집 기능이다. 세션은 Flameshot이나 Ksnip이 제공하지 않는 강력한 기능이다. 해야 할 캡쳐들을 다 하고 난 후 rename하거나, 자체 편집 기능을 사용한 후 원본은 유지하고 편집본만 따로 관리한다던지 하는 것이 편리하게 가능하다.





그런데 최근에 Shutter가 우분투 18.10 이상에서는 관련 라이브러리가 제외되어서 설치가 불가능해졌다. 하지만, 이렇게 유용한 어플리케이션을 누군가가 다시 살려냈다. 이미 우분투 16.04와 루분투 18.04에는 설치를 한터라 테스트는 해보지 못했지만, 추후를 위해 여기에 남겨둔다.

```
sudo add-apt-repository ppa:linuxuprising/shutter
sudo apt-get update
sudo apt install shutter gnome-web-photo
```

1.3.1 편집 기능

루분투 18.04에서 Shutter를 설치하고 나니 편집 버튼이 비활성화 되어 있었다. 구글링 결과를 보고 다음처럼 해주었더니 다시 편집 기능이 사용 가능해졌다.

```
wget https://launchpad.net/ubuntu/+archive/primary/+files/libgoocanvas-common_1.0.0-1_all.deb
sudo dpkg -i libgoocanvas-common_1.0.0-1_all.deb

wget https://launchpad.net/ubuntu/+archive/primary/+files/libgoocanvas3_1.0.0-1_amd64.deb
sudo dpkg -i libgoocanvas3_1.0.0-1_amd64.deb

sudo apt-get install libextutils-depends-perl -y

wget https://launchpad.net/ubuntu/+archive/primary/+files/libgoo-canvas-perl_0.06-2ubuntu3_amd64.deb
sudo dpkg -i libgoo-canvas-perl_0.06-2ubuntu3_amd64.deb
```

1.3.2 Command-line option & i3wm 단축키

Linux 어플 중 많은 수가 명령행 옵션을 제공해준다. GUI로도 기능을 제공해주지만, 명령행 인자로 원하는 기능을 제어할 수 있게 해줌으로써, 원하는 방식으로 사용이 가능하다. Shutter도 마찬가지이다. 제일 많이 사용하는 영역과 전체 화면 캡쳐를 i3wm에서 단축키로 지정해서 사용할 수 있게 해 놓았다.-s는 영역 캡쳐, -f는 전체 화면 캡쳐이다. -d 2는 지연 2초이고, -e는 캡쳐 후 Shutter 세션 윈도우를 표시하지 말라는 옵션이다.

먼저 vi ~/.i3/config 해서 다음을 추가한 후, meta + shift + r 눌러서 i3wm을 재시작해두자.

```
##############################
# Shortcut keys for Shutter #
##############################
# Area
bindsym $mod+Shift+a exec --no-startup-id ~/Data/Works/Private/Utils/i3/shutter_area.sh show
# Full
bindsym $mod+Shift+f exec --no-startup-id ~/Data/Works/Private/Utils/i3/shutter_full.sh show
```

이제는 위에서 지정한 두 개의 스크립트 파일을 생성해줘야 한다.

vi shutter_area.sh해서 다음을 입력한 후 chmod +x shutter_area.sh해서 실행가능하게 만든다.

```
#!/bin/bash
shutter -s -d 2 -c -e
```

shutter_full.sh도 마찬가지로 준비한다.

```
#!/bin/bash
shutter -f -d 2 -c -e
```

이제 meta + shift + a나 meta + shift + f로 바로 바로 캡쳐를 할 수 있다.

1.4 fzf, A command-line Fuzzy Finder

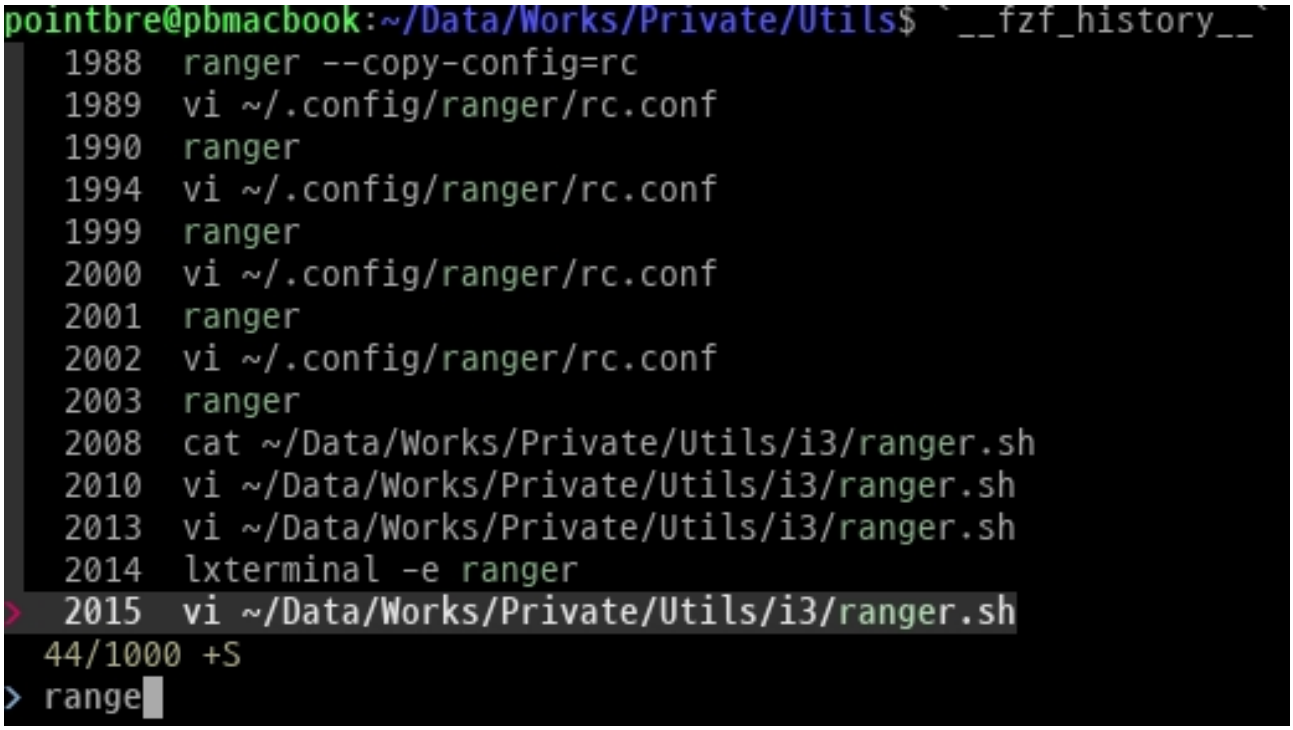
기본적으로 리눅스에서 터미널을 사용하기 시작한 후 ctrl + r을 눌러서 history를 검색하는 기능을 접하게 된다. 그 history 검색을 fuzzy 검색으로 제공해주는 어플리케이션이 fzf(A command-line Fuzzy Finder)이다. 한 2년전 쯤에 설치 후, 설치했다는 사실조차 잊어버릴 만큼, ctrl + r 누를 때마다 도움이 되는, 자그마한 어플리케이션이다. 이 외에도 여러 가지 활용방법이 있지만, 아직까지 본격적으로 사용해보진 않았다. History fuzzy search만 해도 너무 행복하니까.

1.4.1 설치

가장 간단한, 추천할만한 설치 방법은 Linux Brew를 가지고 하는 것이다. 그냥 brew install fzf 하면 설치가 된다. 설치가 끝나면 다음과 같은 출력을 볼 수 있다.

```
==> fzf
To install useful keybindings and fuzzy completion:
  /home/linuxbrew/.linuxbrew/opt/fzf/install

To use fzf in Vim, add the following line to your .vimrc:
  set rtp+=/home/linuxbrew/.linuxbrew/opt/fzf
```

설치 후 /home/linuxbrew/.linuxbrew/opt/fzf/install을 실행하면 된다. 이후 ctrl + r 한 후 몇 글자를 타이핑해보면 다음처럼 아름다운 검색 결과가 나오고, 그 안에서 위 아래로 선택도 가능하다. 기본 제공되는 검색보다 훨씬 편리하다.



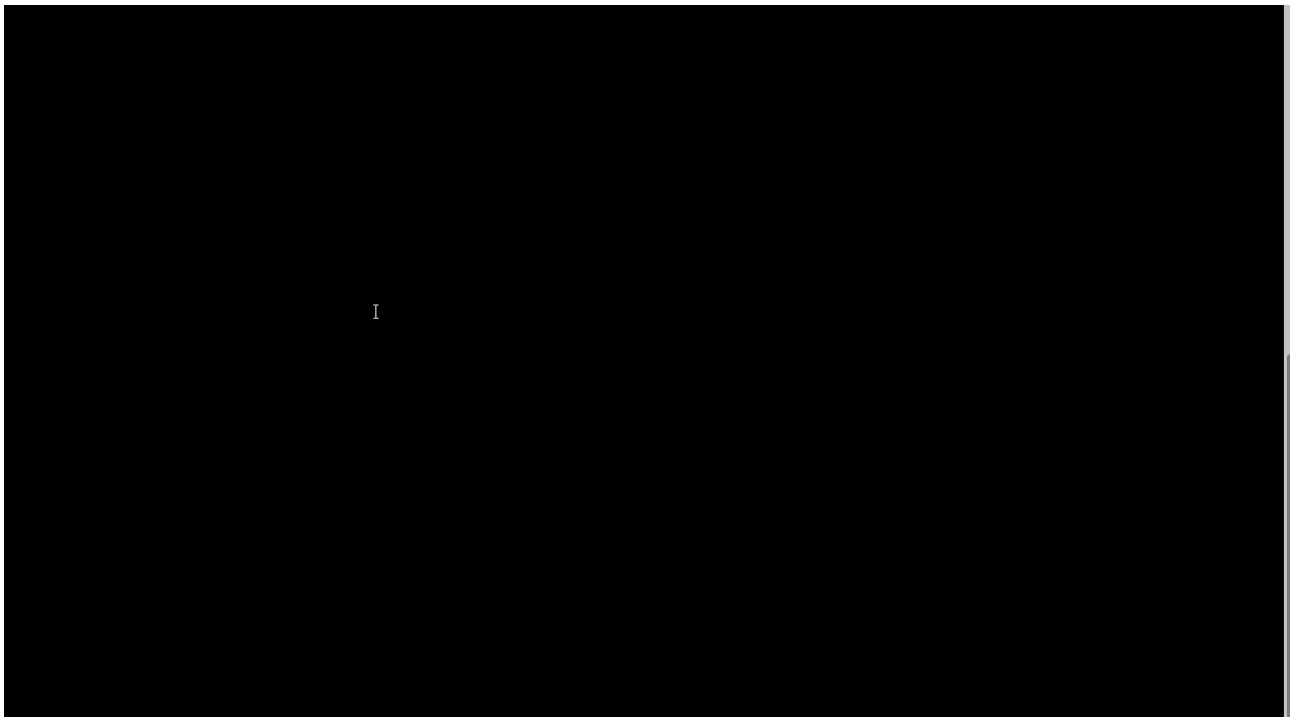
1.5 peek

Screen cast 어플리케이션이다. 단축키(설정에서 변경 가능)로 레코딩 시작/중단이 가능하고, Animated GIF, APNG, WebM, 그리고 MP4 출력을 지원해준다. 가끔 화면을 캡쳐해서 문서에 사용하거나 하는 경우에 매우 유용하다. 다만 Animated GIF를 터미널 화면 캡쳐에 사용시, Animated GIF의 태생적 한계로 인해, 역시나 약간 화질 열화가 느껴진다. 그래서 터미널의 경우는 주로 다른 전용 어플을 쓰는 편이다.

1.5.1 설치

```
sudo add-apt-repository ppa:peek-developers/stable
sudo apt update
sudo apt install peek
```

1.5.2 샘플



1.6 asciinema + asciicast2gif + svg-term

Peek는 주로 터미널 이외에 사용하고 터미널 캡처에는 asciinema와 변환 툴 2개를 사용하는 중이다.

1.6.1 설치

Imagemagick이 설치되어 있다면(display --version 해서 잘 실행되면 설치되어 있는 것이다.) 다음만 설치해주면 된다.

```
brew install asciinema gifsicle
npm install -g asciicast2gif
npm install -g svg-term-cli
```

1.6.2 녹화하기

asciinema rec asciinema.json 해서 레코딩 시작 후 exit이나 ctrl + d를 눌러서 레코딩을 종료한다. 웹사이트에 등록할 의도는 없으므로 다음처럼 animated GIF나 SVG animation으로 만들어서 문서에 첨부하거나, 웹사이트에 추가하려고 한다.

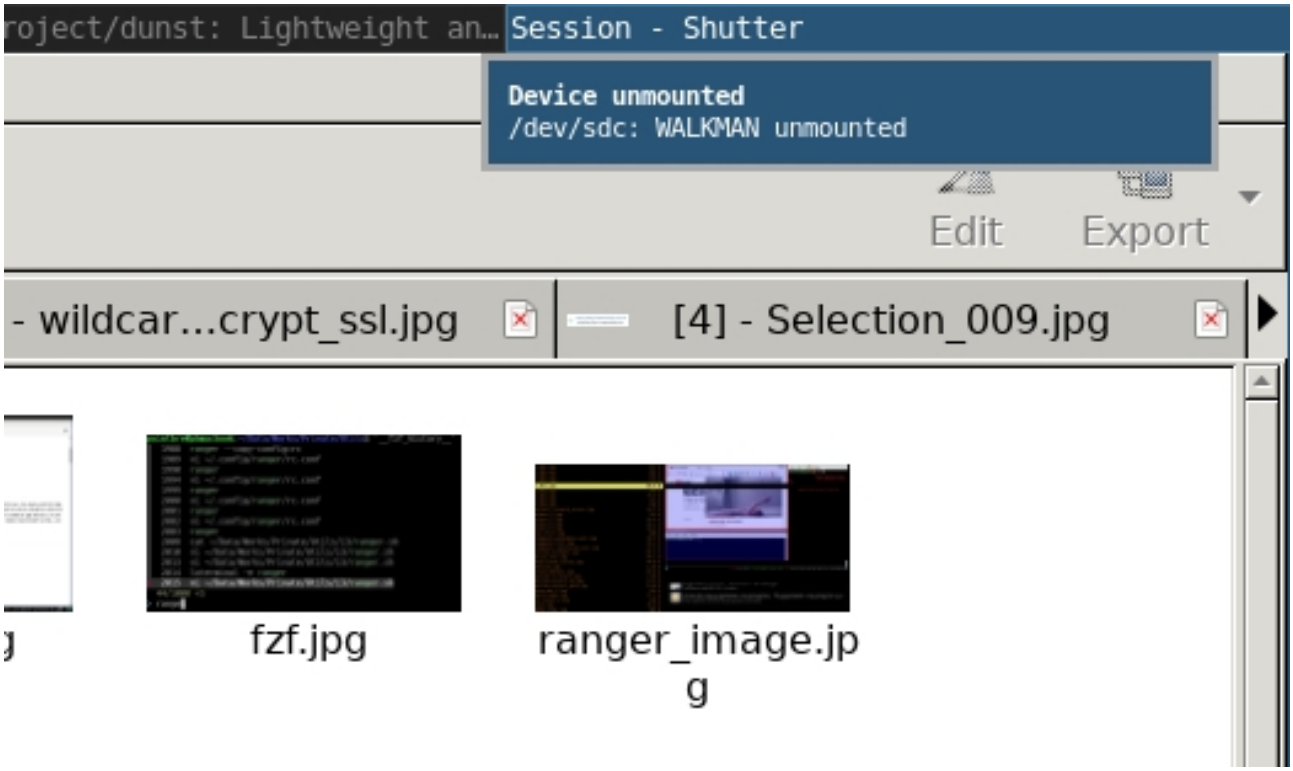
1.6.3 Animated GIF

asciicast2gif -w 80 -h 20 asciinema.json asciinema.gif을 실행해서 asciiena를 가지고 생성한 결과물을 GIF로 변환할 수 있다. -w 80은 80 컬럼, -h 20은 20 라인을 의미한다. 다음 결과물을 보면 상당히 깔끔하다는 걸 알 수 있다.

SVG animation

svg 애니메이션으로 변경하려면 cat asciinema.json | svg-term --width 80 --height 20 > asciinema.svg 처럼 변환한 후 브라우저로 열어서 확인해보면 된다. 참고로 --width와 --height는 픽셀이 아니라, 컬럼과 라인 개수이다. 벡터 기반 animation이라서 제일 깔끔하지만, 브라우저 말고는 제대로 지원해주는 곳이 별로 없어서 주로 웹문서에 추가하는 경우에 사용중이다.

1.7 dunst



i3wm에 어울리는 notification daemon을 찾는 중 얻어 걸린 녀석이다. 다양한 설정이 가능하지만, 무엇보다도 메시지에 기반한 자동화가 가능하다는 점이 가장 큰 장점이라고 생각된다.

1.7.1 설치

최신 버전을 원한다면

소스를 다음처럼 빌드해야 한다.

```
mkdir -p ~/projects/dunst
cd ~/projects/dunst
git clone https://github.com/dunst-project/dunst.git .
make
sudo make install
```

이제 dunst -v 하면 Dunst - A customizable and lightweight notification-daemon v1.3.2-110-ga0f21f5처럼 설치된 버전 정보를 알 수 있다. 혹시 빌드가 실패하면 dependency 목록을 참조해서 해결해야 한다.

최신 버전이 아니여도 좋다면

sudo apt install -y dunst처럼 그냥 설치 가능하다. 다만 설정 파일인 dunstrc의 위치가 좀 다르다.

1.7.2 Notification daemon을 dunst로 변경

which dunst 실행해서 dunst 설치 위치를 찾는다. 우분투 16.04의 경우 /usr/local/bin/dunst였다. 이제 데몬 설정 파일을 변경해야 한다. sudo vi /usr/share/dbus-1/services/org.freedesktop.Notifications.service 실행한 후 Exec=... 라인을 Exec=/usr/local/bin/dunst로 변경한다.

1.7.3 설정파일 복사하기

설치시에 기본 dunstrc 파일이 /usr/local/share/dunst/dunstrc 또는 /etc/xdg/dunst/dunstrc에 복사되니까 그걸 ~/.config/dunst/dunstrc로 복사해서 개인별 설정을 추가/변경하면 된다.

```
mkdir -p ~/.config/dunst
sudo cp /usr/local/share/dunst/dunstrc ~/.config/dunst/dunstrc
CURR_USER=`id -un $USER`
```

```
CURR_GROUP=`id -gn $USER`
sudo chown $CURR_USER:$CURR_GROUP ~/.config/dunst/dunstrc
```

1.7.4 메시지 테스트

notifi-send가 설치되지 않은 상황이라면, sudo apt install libnotify-bin 해서 설치 후 테스트하면 된다.

```
notify-send --urgency=low "Summary 1" "Body 1"
notify-send --urgency=normal "Summary 2" "Body 2"
notify-send --urgency=critical "Summary 3" "Body 3"
```

실행하면 low 등급은 검정색 배경, normal은 파란 배경, critical은 빨간 배경 박스에 메시지가 출력되는 것을 보게 된다.

1.7.5 Notification 메시지 수신시 스크립트 실행하기

바로 이 기능이 Dunst를 선택할 수 밖에 없었던 중요한 기능이다. Notification 메시지 패턴에 따라 원하는 쉘스크립트나 어플리케이션을 실행할 수 있는데, 이를 이용해서 여러가지를 나중에 응용할 예정이다.

~/.config/dunst/dunstrc 변경하기

다음을 추가해준다.

```
[script-test]
   summary = "*script:hello.sh*"
   script = ~/bin/hello.sh
```

hello.sh 만들기

```
mkdir ~/bin
cd ~/bin
touch hello.sh
chmod +x hello.sh
```

vi hello.sh 한 후 다음처럼 입력해준다.

```
#!/bin/bash
notify-send "Hello" "Hello received from android phone"
```

dunst process 종료해서 재시작시키기

dunstrc가 변경되었을 때는 killall dunst 해서 실행 중인 dunst를 종료해야 반영이 된다.

테스트

```
notify-send --urgency=normal "script:hello.sh" "Running hello.sh through notification message"
```

이제 Notification 메시지가 도착하고 나서 다시 Hello 메시지가 표시될 것이다.

조금 더 깊숙히

hello.sh를 다음처럼 수정한 후 다시 테스트해보자.

```
#!/bin/bash
echo $@ >> /tmp/test.log
```

notify-send --urgency=normal "script:hello.sh" "Running hello.sh through notification message" 실행해서 메시지 보낸 후 /tmp/test.log를 열어보면 다음과 같은 명령행 인자들이 hello.sh에게 전달된 것을 알 수 있다. 즉, 보낸 메시지가 다음처럼 그대로 스크립트에 전달이 되고, 그것에 기반해서 스크립트가 할 동작을 결정할 수 있다는 것이다.

```
notify-send script:hello.sh Running hello.sh through notification message dialog-information NORMAL
```

1.8 udiskie

사진 찍은 후 micro sd 카드를 리눅스 pc에 꼽기만 하면 자동으로 Google Photo로 업로드 후, 업로드 제대로 되었는 지 확인 후, 카드의 용량을 비워주고 Pushbullet 메시지를 보내주는 자동화를 구현하다가 발견한 어플이다. 윈도우에서 장치 제거 관리자 같은 기능이라고 생각하면 된다. sd 카드 등이 끼워졌을 경우 자동으로 마운트 해주며, 시스템 트레이의 아이콘을 클릭하면

기기별로 Unmount나 Eject 등의 관리가 가능하다. 다른 점이라면, 시스템 메시지를 보내준다는 점이다. 바로 그 점에 착안해서, mount 되었다는 메시지에 반응해서 실행되는 쉘 스크립트로 구글 포토로 업로드하려고 했다. 문제가 좀 생겨서 무한정 연기 중이긴 하지만, 언젠가는 공유할 수 있을 것 같다. 언젠가는...

1.8.1 설치

구글링에 나온 가이드를 참조했다. 그런데 Ubuntu 16.04와 18.04에서는 udisks가 아니라 udisks2를 설치해야 하므로, 가이드에 나온 dependency 목록에서 udisks를 udisks2로 바꿔서 다음처럼 dependency들을 설치해준다.

```
sudo apt-get install -y python-setuptools udisks python-pip python-gobject python-yaml libgio2.0 gobject-introspection
↪ libgtk2.0-0 libnotify4 gettext gir1.2-notify-0.7
sudo pip install udiskie
```

1.8.2 Policy kit 설정

sudo vi /etc/polkit-1/localauthority/50-local.d/consolekit.pkla 한 후 다음을 추가해준다.

```
[mount]
Identity=unix-group:plugdev
Action=org.freedesktop.udisks.filesystem-mount
ResultAny=yes
[unlock]
Identity=unix-group:plugdev
Action=org.freedesktop.udisks.luks-unlock
ResultAny=yes
[eject]
Identity=unix-group:plugdev
Action=org.freedesktop.udisks.drive-eject
ResultAny=yes
[detach]
Identity=unix-group:plugdev
Action=org.freedesktop.udisks.drive-detach
ResultAny=yes
```

1.8.3 plugdev 그룹

groups 해서 plugdev 그룹에 속해있는지 확인한다. 안되어 있다면 sudo usermod -a -G plugdev $USER 해서 추가해준다.

1.8.4 실행용 스크립트

vi udiskie.sh 한 후 다음을 작성해 넣는다. --automount는 말 그대로 자동 마운트를 활성화 하는 것이고, --notify는 이벤트 발생시 시스템 알림 메시지를 발송하게 하는 옵션이다. --tray는 시스템 트레이에 아이콘을 표시하게 하고, 마지막으로 --use-udisks2는 udisk1이 아니라 udisk2(잘 몰라도 된다!)를 사용하게끔 설정하는 것이다.

```
#!/bin/bash
udiskie --automount --notify --tray --use-udisks2
```

1.8.5 부팅 후 자동 시작

i3

vi ~/.config/i3/config 한 후 exec --no-startup-id ~/Sync/Utils/i3/udiskie.sh처럼 위에서 생성한 스크립트를 지정하면 된다. 재부팅 후 sd 카드를 넣자 자등으로 마운트되었고 notification 메시지도 잘 도착했다.

LXDE

기본 설정 - LXSession 기본 프로그램 - 자동 시작에서 직접 추가하거나 ~/.config/lxsession/Lubuntu/autostart에 기재하면 되는데, 현재는 autostart 파일을 이용하는 중이다. nohup sh -c "/path/udiskie.sh" &라고 추가한 후 재로그인해서 udiskie가 실행되는지 확인해본다.

1.8.6 PCMan File Manager의 자동 마운트 기능 해제

이 파일 매니저를 사용중이라면 Edit - Preference - Volume Management에서 자동 마운트 관련 옵션을 모두 꺼서, udiskie가 대신 할 수 있도록 해줘야 한다.

1.9 xtail

리눅스 사용하기 시작하면서 제일 먼저 애타게 찾았던 어플이다. 주로 터미널 작업을 하면서, 어플리케이션의 실행 로그를 감시하거나 하는 일이 많다. 그런데 tail은 향후에 생성될 파일에 대해 사용이 불가능하다. 그래서 watch등과 조합해서 사용하다가 이 녀석을 알게 되고 나서는 모든 게 해결되었다. 그냥 xtail dir1/*.LOG dir2/*.LOG 처럼 모니터링하려고 하는 파일 패턴만 기재하면, 향후 생성되는 파일에 대해서도 tail을 해준다. 개인적으로는 정말 유용하게 사용중이다. 참고로 빠져나올 때는 ctrl + backslash(₩)을 눌러야 한다. 설치하려면 sudo apt install -y xtail 하면 되는 데, 하도 오래된 유틸이라 대부분의 버전에서 사용 가능하다.

1.10 fswatch + git-sync

1.10.1 fswatch

파일이나 디렉토리의 변경을 감지해주는 어플리케이션이다. 대상 파일/디렉토리가 변경될 때 어떤 일을 하고 싶은 경우에 유용하다. Ubuntu 18.04용으로 fswatch가 제공되지만 버전이 오래된 버전이다. 최신 버전을 사용하기 위해서 Brew를 사용해서 설치하는 것이 좋다. brew install fswatch을 실행하면 된다.

1.10.2 git-sync

몇 시간 투자하고 테스트해 본 후 최종 정착한, 개발된지 오래된 shell script이다. 직접 테스트해 보니, 별 다른 문제없이 잘 동작을 해서, 몇 달째 잘 사용중이다. 기본적으로 remote repo의 변경사항을 pull 먼저하고, local repo에 있는 변경사항을 add 하고 commit한 후 push까지 해 준다. 실제로 구현하려고 해 봤는데 다양한 문제에 부딪혀 포기했었었다. 하지만 이 쉘스크립트는 어쩐 일인지 잘 동작한다. 프로젝트 페이지에 가면 최신판을 다운로드 할 수 있다.

1.10.3 자동 git 싱크

다음은 Notable로 작성한 Markdown 파일들을 Remote git repository로 자동으로 싱크하기 위해 간단히 작성한 스크립트이다. flock을 이용해서 한 번에 한 개의 싱크가 이뤄지게 한 점이 특징이다.

```
#!/bin/bash

cd /path/to/git_local_repo
fswatch -o . | while read num; do (flock -x 200 && gitsync -m "saved" && sleep 60) 200> /tmp/notes.lock ; done &
```