토막 상식

Nvidia Driver 설치후에도 적용이 안되는 경우 확인

현재 드라이버 버전 확인 및 조치
로그인 가능한 다른 VT로 전환 후 다음과 같이 진행

```
nvidia-smi --version | grep -i driver
DRIVER version      : 590.48.01
sudo -s
mkdir -pv /usr/lib/firmware/nvidia/590.48.01
cp /usr/src/NVIDIA-Linux-x86_64-590.48.01/firmware/* \
   /usr/lib/firmware/nvidia/590.48.01
modprobe -r -f nvidia
modprobe nvidia
systemdctl restart sddm
```

Meson Bash-completion 수정

meson에서 환경변수가 tab completion이 안되는 경우.
/usr/share/bash-completion/completions/meson

```
#complete -F _meson meson
complete -o default -o bashdefault -F _meson meson
```

Bash 토막 상식

```
# 인자가 비어있으면 미리 지정된 값 할당
TARGET_ARCH=${LFS_TARGET:-x86_64-lfs-linux-gnu}
# 인자 내용에서 확장자 제거
FILE="ffmpeg-8.0.1.tar.xz"
DIR_NAME=${FILE%.tar*} # DIR_NAME="ffmpeg-8.0.1"
# 변수 내용 치환
SAFE_NAME=${RAW_NAME// /_} # " " -> "_"
# 경로인 경우 조작
# ## 왼쪽부터 가장 먼 "/" 까지 삭제. 두번 사용해서 최대한 길게 적용.
# % 오른쪽부터 처음 나온 "/" 까지 삭제. 두번 사용하면 가장 왼쪽의 "/" 까지 삭제
PATH="/usr/lib/pkgconfig/sdl3.pc"
echo ${PATH##*/}  # 마지막 / 앞부분을 다 날림 -> sdl3.pc
echo ${PATH%/*}   # 마지막 / 뒷부분을 다 날림 -> /usr/lib/pkgconfig
```

pkgconf로 설치된 라이브러리 확인

Bash Script

```
### Check installed packages
TARGET=""

### merge args
for UARGS in "$@"; do
        TARGET="${TARGET}|${UARGS}"
done

### remove first "|'
TARGET="${TARGET#|}"

### '-print0' and 'xargs -0' for file name with white space
find /usr/lib/pkgconfig -name "*.pc" | \
        grep -Ei "${TARGET}" | \
        xargs -0 pkgconf --print-provides
```

Ruby Script

```
#!/usr/bin/env ruby

# 1. 인자 처리
target = ARGV.empty? ? "." : ARGV.join('|')

# 2. 파일 목록 가져오기 및 패키지명 추출
search_path = "/usr/lib/pkgconfig/*.pc"

# .pc 확장자를 제외한 파일 이름(패키지명)만 추출하여 필터링
package_names = Dir.glob(search_path).map do |file|
  File.basename(file, ".pc") # "/usr/lib/pkgconfig/zlib.pc" -> "zlib"
end.select do |pkg|
  pkg =~ Regexp.new(target, Regexp::IGNORECASE)
end

# 3. pkgconf 실행
if package_names.any?
 # 패키지명을 나열하여 실행
 # 예: pkgconf --print-provides zlib libpng
  system("pkgconf", "--print-provides", *package_names)
else
  puts "매칭되는 패키지가 없습니다."
end
```

마이크로코드 Grub auto detect

AMD 5800X 기준

```
sudo -s
cp /usr/lib/firmware/amd-ucode/microcode_amd_fam19h.bin /boot/amd-ucode.img
grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
```

```
Found theme: /usr/share/grub/themes/starfield/theme.txt
Found linux image: /boot/vmlinuz-6.18.10-1st
Found initrd image: /boot/amd-ucode.img
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
```

설치된 Perl Module 목록 확인

Perl과 기본 시스템 도구만 이용

```
#!/bin/sh

# Begin ~/bin/pml.sh

# List of installed Perl Modules

# grep "$pattern" - $1이 비어있으면 전체 검색(.), 있으면 $1로 검색
# /"Module"/ - "Module" 가 포함된 줄만 선택
# !seen[$NF]++ - awk 내부 배열을 사용해서 중복 제거
# print $NF - 마지막 단어(모듈명)
# sort --ignore-case (-f) | 대소문자 구분없이 정렬

pattern="${1:-.}"

perldoc perllocal | awk '/"Module"/ {if (!seen[$NF]++) print $NF}' | sort --
ignore-case | grep --color=always -E -i "$pattern"

# End ~/bin/pml.sh
```

```
pml.sh "tiny|test"
Capture::Tiny
Class::Tiny
Test::Deep
Test::Exception
Test::Fatal
Test::Needs
Test::Warnings
Try::Tiny
```

이런 방식으로 다중 검색도 가능함.

ExtUtils::Install module 이용

```
#!/bin/sh

# Begin ~/bin/pml.sh

# List of installed Perl Modules

# 검색어가 없으면 사용법 출력
if [ -z "$1" ]; then
        echo "Usage: $(basename $0) [pattern]"
        echo "Example: $(basename $0) build"
        exit 1
fi

QUERY=$1

echo "---- Searching installed Perl modules for: $QUERY ----"

# ExtUtils::Installed를 사용하여 설치된 모듈 목록과 버전을 출력
# grep -i를 사용하여 대소문자 구분 없이 검색

perl -MExtUtils::Installed -e '
        my $inst = ExtUtils::Installed->new();
        foreach my $mod ($inst->modules()) {
                my $ver = $inst->version($mod) || "N/A";
                print "$mod | Version: $ver\n";
        }
' | grep --color=auto -E -i "$QUERY"

# 일치하는 결과가 없으면

if [ ${PIPESTATUS[1]} -ne 0 ]; then
        echo "Result: No Matching Modules found."
fi

# End ~/bin/pml.sh
```

```
pml.sh text
---- Searching installed Perl modules for: text ----
Text::BibTeX | Version: 0.91
Text::Diff | Version: 1.45
Text::Glob | Version: 0.11
```

버전을 출력해주는 장점이 있음.
특정 모듈 설치전에는 사용하기 힘들다는 단점이 있음.

설치된 Python3 Module 목록 확인

```
pip3 list
```

Qemu Screen Dump

1. <Ctrl>+<Alt>+<2> 그래픽 모드
   <Ctrl>+<A>, <C> -nographic 모드
   모니터 콘솔로 전환
2. screendump filename.ppm
   다음과 같이 **-f** 옵션으로 저장 포맷 지정 가능
   screendump screenshot.png -f png
   포맷 지정 저장 방식에서 세그폴트 발생하면 ppm으로 저장후 변환
3. <Ctrl>+<Alt>+<1> 가상 머신 화면 복귀

xwindow screen saver & dpms 시간 설정

```
xset s 600 600
xset dpms 600 600 600
```

순서대로
화면보호기 작동시간,전환시간
DPMS 대기,유예,끄기

for, echo and sed

```
for i in $(echo "ldapadd, ldapcompare, ldapdelete, ldapexop, ldapmodify,
slapschema, slaptest" | sed 's/,/ /g') ; do find /usr/bin -type f -name $i ;
done
```

Grep && Tar

특정 문자열을 가진 파일을 아카이브 처리

“python3_11” 문자열이 있는 파일을 검색하고 $HOME에 아카이브 파일 생성

```
grep -rl python3_11 | tar -T - -acf ~/ibus-hangul.tar.zst
```

grep

- r 재귀 탐색
- l 파일명을 상대경로로 출력

tar

- T 입력된 파일에서 아카이브 대상 불러옴
- - 파이프로 전달받은 표준 입력

- a 확장자로 압축 방법 자동 선택
- c 아카이브 생성
- f 아카이브 파일명

Configure Options #1

사용해본 결과 생각처럼 편한것은 아니었음.

패키지마다 다르지만 사용하지 않는 옵션이 주어지면 Fail이 발생하는 경우가 있었음.

-docdir=/usr/share/doc/<package name> 입력이 귀찮아서 해보려고 했지만

결과적으로는 그냥 타이핑 하는것이 적합했음.

내용 확인

$HOME/.bashrc에 추가

```
if [ -d $HOME/.bashrc.d ] ; then
    for conf in $HOME/.bashrc.d/*.conf ; do
        source $conf
    done
fi
```

```
cat > $HOME/.bashrc.d/config_opts.conf << EOF
alias doc-dir='echo "--prefix=/usr --enable-shared --disable-static --
sysconfdir=/etc --docdir=/usr/share/doc/$(basename $(pwd)) --enable-
dependency-tracking"'
EOF
```

```
./configure $(doc-dir)
checking build system type... x86_64-pc-linux-gnu
checking host system type... x86_64-pc-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
... ... 중략 ... ...
checking that generated files are newer than configure... done
configure: creating ./config.status
config.status: creating Makefile
config.status: creating src/Makefile
config.status: creating tests/Makefile
config.status: creating config/Makefile
config.status: creating config/gdlib.pc
config.status: creating src/config.h
config.status: executing depfiles commands
config.status: executing libtool commands
```

```
make
sudo make instasll
```

Find script

위쪽에 Bash & Ruby 스크립트 사용 패키지 설치 여부가 생각나지 않을 때 사용

필요 패키지: ANSI Code Generator
git clone https://github.com/fidian/ansi.git

사용 예시

```
baecy-lfs@m5pro-lfs < ~ > $
double-check libGLEW
/usr/lib/libGLEW.so
/usr/lib/libGLEW.so.2.2
/usr/lib/libGLEW.so.2.2.0
I searched for libGLEW you wanted in th /usr you specified
baecy-lfs@m5pro-lfs < ~ > $
```

```
function file-finder() {
    local FindWord=*${1}*
    local FindDir=$2
    if [ -z $FindDir ] ; then
        FindDir=/usr
    fi
    find $FindDir -name $FindWord 2> /dev/null | sort | grep -E --
color=always $1
    echo "Found file $(ansi --green ${1}) in directory $(ansi --yellow
${FindDir}) (refer to the list abovc)."
}
export -f file-finder
```

Patch with Wget

```
wget -q -O- <patch url> | patch <options>
```

사용법

예시 1

docbook-xsl-nons-1.79.2 patch

```
wget -q -O-
https://www.linuxfromscratch.org/patches/blfs/12.1/docbook-xsl-nons-1.79.2-s
```

```
tack_fix-1.patch \
| patch -Np1
```

예시 2

libpng patch

```
wget -O-
https://downloads.sourceforge.net/sourceforge/libpng-apng/libpng-1.6.40-apng
.patch.gz \
| gzip -cd | patch -p1
```

PS1 설정

일반사용자

```
\[\033[01;34m\]\u\[\033[00m\]@\[\033[01;36m\]\h
\[\033[01;34m\]\w\[\033[00m\] \$
```

루트사용자

```
\[\033[01;35m\]\u\[\033[00m\]@\[\033[01;35m\]\h \[\033[01;37m\]\w
\[\033[01;31m\]# \[\033[00m\]
```

2차 개선

별거아닌데 쓸데없이 길어짐

```
PS1_ROOT='\[$(tput setaf 203)\][ \[$(tput sgr0)\]\w \[$(tput setaf 203)\]]
\[$(tput sgr0)\]\$\n'
PS1_USER='\[$(tput setaf 75)\][ \[$(tput sgr0)\]\w \[$(tput setaf 75)\]]
\[$(tput sgr0)\]\$\n'
PS1_HEAD='\[$(tput setaf 229)\]\u\[$(tput setaf 199)\]@\[$(tput setaf
215)\]\h '
if [ "$(awk '$5=="/" {print $1}' </proc/1/mountinfo)" != "$(awk '$5=="/"
{print $1}' </proc/$$/mountinfo)" ] ; then
    if [ $(id -u) -eq 0 ] ; then
        PS1="<Chroot>${PS1_HEAD}${PS1_ROOT}"
    else
        PS1="<Chroot>${PS1_HEAD}${PS1_USER}"
    fi
else
    if [ $(id -u) -eq 0 ] ; then
        PS1="${PS1_HEAD}${PS1_ROOT}"
    else
        PS1="${PS1_HEAD}${PS1_USER}"
    fi
```

```
fi
```

1차 개선

번거로운건 사라짐 그러나 여전히 지저분함

```
# chroot 환경 확인
if [ "$(awk '$5=="/" {print $1}' </proc/1/mountinfo)" != "$(awk '$5=="/"
{print $1}' </proc/$$/mountinfo)" ] ; then
   PS1='<Chroot> \[$(tput setaf 229)\]\u\[$(tput setaf 199)\]@\[$(tput setaf
215)\]\h \[$(tput setaf 203)\][ \[$(tput sgr0)\]\w \[$(tput setaf 203)\]]
\[$(tput sgr0)\]\$\n'
else
# root 사용자 확인
        if [ $(id -u) -eq 0 ] ; then
                PS1='\[$(tput setaf 229)\]\u\[$(tput setaf 199)\]@\[$(tput
setaf 215)\]\h \[$(tput setaf 203)\][ \[$(tput sgr0)\]\w \[$(tput setaf
203)\]] \[$(tput sgr0)\]\$\n'
        else
                PS1='\[$(tput setaf 229)\]\u\[$(tput setaf 199)\]@\[$(tput
setaf 215)\]\h \[$(tput setaf 75)\][ \[$(tput sgr0)\]\w \[$(tput setaf
75)\]] \[$(tput sgr0)\]\$\n'
        fi
fi
```

최초

많이 지저분하고 번거로움

```
touch $LFS/etc/lfs-chroot
```

~/.bashrc

```
if [ -f /etc/lfs-chroot ]; then
  PS1='\[$(tput setaf 75)\]<LFS> \[$(tput setaf 229)\]\u\[$(tput setaf
199)\]@\[$(tput setaf 215)\]\h \[$(tput setaf 75)\]\w \[$(tput sgr0)\]\$\n'
else
  PS1='\[$(tput setaf 75)\]\[$(tput setaf 229)\]\u\[$(tput setaf
199)\]@\[$(tput setaf 215)\]\h \[$(tput setaf 75)\]\w \[$(tput sgr0)\]\$\n'
fi
```

Debian 설치 후 추가적인 사항

LFS 진행에 필요한 패키지 설치

```
sudo apt install build-essentials bison gawk m4 texinfo texinfo  ## 필수 사항
sudo apt install gettext libisl-dev                              ## 선택 사항
sudo ln -sf bash /usr/bin/sh && file /usr/bin/sh                 ## 필수 사항
/usr/bin/sh: symbolic link to bash
```

화면 출력 로그파일로 저장

```
time { command 1 && ... && command N; } 2>&1 | tee <log-file>
```

Qemu에서 부팅중 마운트 에러가 발생하는 경우

Qemu에서 가상머신 시작시 디바이스 순서가 바뀌는 일이 발생해서 처리한 내용.

파티션 레이블 설정

```
mkswap -L <label> <partition>
e2label <device> <label>
```

/etc/fstab 수정

```
# /etc/fstab: static file system information.
#
# Use 'blkid' to print the universally unique identifier for a
# device; this may be used with UUID= as a more robust way to name devices
# that works even if disks are added and removed. See fstab(5).
#
# systemd generates mount units based on this file, see systemd.mount(5).
# Please run 'systemctl daemon-reload' after making changes here.
#
# <file system>         <mount point>   <type>          <options>
<dump>  <pass>
# / was on /dev/sda1 during installation
LABEL=ROOT_DISK          /              ext4            errors=remount-ro
0       1
# swap was on /dev/sda5 during installation
LABEL=SWAP_1G           none            swap            sw
0       0
LABEL=SWAP_5G           none            swap            sw
0       0
# /dev/sr0              /media/cdrom0   udf,iso9660     user,noauto
```

```
0       0
LABEL=LFS_DISK          /mnt/lfs        ext4            defaults
1       1
```

UUID로 설정하는 경우 장치명을 주석으로 표기하면 추후 수정시 용이.

Putty에서 Ncurses 설치 후 <Home>,<End> Key 사용 설정

~/.profie or ~/.bashrc

```
TERM=putty-256color
export TERM
```

Glibc-2.35 Compile

컴파일중 알 수 없는 에러가 발생하고 매번 발생 위치가 다르다면 -j1 인자로 병렬처리 없이 make 실행

```
make -j1
```

LFS로 Booting후 ''su'' 명령이 안되는 경우

*root*로 다음과 같이 실행

```
chmod 4755 /usr/bin/su
```

su 파일에 setuid를 설정해서 문제를 해결.

LFS로 부팅하기 전에 준비할 것들

OpenSSH

OpenSSH

Wget

소스 패키지 다운로드에 필요 Wget-1.24.5

설치 순서 : Libunistring - Libidn2 - Libpsl - Libtasn1 - P11-kit - SQLite - NSPR - NSS -Make-ca - Wget

의존성

- Libpsl-0.21.5 REQ
  - Libunistring-1.2 REQ
  - Libidn2-2.3.7 REQ

        - Libunistring-1.2 REQ
- Make-ca-1.13 Runtime REQ
    - Libtasn1-4.19.0 REQ
    - P11-kit-0.25.3 REQ
        - Libtasn1-4.19.0 REQ
        - Make-ca-1.13 Runtime REQ
    - NSS-3.99 Runtime REQ
        - NSPR-4.35 REQ
        - SQLite-3.45.3 REC
        - P11-kit-0.25.3 Runtime REC

NFS-UTILS

NFS-Utils-2.6.4

필요한 패키지를 아래의 순서대로 설치 후 설치

의존성

- libtirpc-1.3.4 REQ
- libevent-2.1.12 REQ
- rpcsvc-proto-1.4.4 REQ
- SQLite-3.45.1 REQ
- rpcbind-1.2.6 REQ

SSHFS (NFS 사용 안하는 경우)

N40L에 있는 소스 디렉토리 마운트에 필요

sshfs-3.7.3

설치 순서 : ICU - Libxml2 - Sgml-common - UnZip - Docbook-xsl-nons - Docbook-xml - Libxslt - Docutil - Packing - PCRE2 - Glib - OpenSSH - Fuse - SSHFS

의존성

- Fuse-3.16-1 REQ
- Glib-2.80.0 REQ
    - Packging (Python module REQ
    - Docutils (Python module REC
    - Libxslt-1.1.39 REC
        - Libxml2-2.12.6 REQ
            - ICU-75.1 REC
        - Docbook-xml-4.5 Runtime REC
            - Libxml2-2.12.6 REQ
            - Sgml-common-0.6.3 REQ
            - UnZip-6.0 REQ
        - Docbook-xsl-nons-1.79.2 Runtime REC
            - Libxml2-2.12.6 REQ

    - Pcre2-10.43 REC
- OpenSSH 9.6p1 REQ

Textinfo Dir File Rebuild

```
pushd /usr/share/info
  rm -v dir
  for f in *
    do install-info $f dir 2>/dev/null
  done
popd
```

Swap file

```
export LFS=/mnt/lfs
export LFS_SWAP=$LFS/Swapfile_For_LFS_1G
sudo fallocate --length 1G $LFS_SWAP
sudo chmod 0600 $LFS_SWAP
sudo mkswap $LFS_SWAP
Setting up swapspace version 1, size = 1024 MiB (1073737728 bytes)
no label, UUID=890ba9a5-da48-4374-84ce-b71b91863e00
sudo swapon $LFS_SWAP
sudo echo "$LFS_SWAP  swap  swap  defaults  0  0" >> /etc/fstab
```

Binutils 설치하기 전에 ISL 설치하기

GMP를 먼저 설치

```
./configure --prefix=/usr    \
            --enable-cxx     \
            --disable-static \
            --docdir=/usr/share/doc/gmp-6.2.0
make
make check 2>&1 | tee gmp-check.log
awk '/# PASS:/{total+=$3} ; END{print total}' gmp-check-log
```

ISL Homepage에서 받아서 준비

```
./configure --prefix=/usr --disable-static --with-gmp=system
make
make install
```

$LFS/sources에서 디렉토리 검색

가장 간단한 방법

```
ls -d */
```

```
alias dirfind="find -mindepth 1 -maxdepth 1 -type d | sed 's@^./@@'"
dirfind | wc -l    ## 디렉토리 갯수 확인
rm -rf $(dirfind)  ## 설치 완료된 소스 디렉토리 삭제
## 설치 완료된 패키지 디렉토리만 삭제할거면 다음과 같이
alias SearchAndDestroy='find -mindepth 1 -maxdepth 1 -type d -exec rm -rf {}
\;'
```

SBU 측정

```
## 현재 디렉토리+생성시간을 추가해서 로그파일 작성
alias lfslog='tee ~/lfs-log/$(case $(basename $(pwd)) in build) echo
$(basename $(dirname $(pwd))); ;; *) echo $(basename $(pwd)); ;;
esac).$(date "+%Y%m%d_%H%M%S").log'
time { ./configure .... && make && make install; } | lfslog
```

짜투리

```
<a href=""></a>                                                    ## 현재 창 또는 탭에서
링크 열기
<a target="_blank" rel="noopener noreferrer" href=""></a> ## 새로운 창 또는 탭에
서 링크 열기
```

```
./configure --prefix=/usr        \
            --bindir=/usr/bin    \
            --localstatedir=/var \
            --disable-logger     \
            --disable-whois      \
            --disable-rcp        \
            --disable-rexec      \
            --disable-rlogin     \
            --disable-rsh        \
            --disable-servers

## Same Command
./configure --prefix=/usr --bindir=/usr/bin --localstatedir=/var \
            --disable-{logger,whois,r{cp,exec,login,sh},servers}
```

tar를 이용한 복사

```
tar -cf - . | tar -xvf - -C /target_directory
```

find example

```
## 크기가 50M 이상이면서 .deb, .vmdk 확장자가 아닌 파일
## Operator AND = -and,-a, OR= -or = -o, NOT = !
find /media/d/ -type f -size +50M -and ! -name "*deb" -and ! -name "*vmdk"
```

grep example

```
## 일반적인 grep 으로 다음과 같은 결과가 나왔다.
$ grep 'printf' fileio.c
                printf("FILE open Error\n");
                printf("chi = %c\n", chi);
                printf("cho = %c\n", cho);
                printf("FILE open Error\n");
                printf("chi = %c\n", chi);
                printf("cho = %c\n", cho);

## -n 옵션을 통해 line 을 볼수 있다.
$ grep -n 'printf' fileio.c
11:             printf("FILE open Error\n");
18:             printf("chi = %c\n", chi);
19:             printf("cho = %c\n", cho);
25:             printf("FILE open Error\n");
31:             printf("chi = %c\n", chi);
32:             printf("cho = %c\n", cho);

## -v 옵션을 통해 'chi' 문자열이 들어간걸 제외하고 print 했다.
$ grep -n 'printf' fileio.c | grep -v 'chi'
11:             printf("FILE open Error\n");
19:             printf("cho = %c\n", cho);
25:             printf("FILE open Error\n");
32:             printf("cho = %c\n", cho);

## grep 을 두번 사용해서 보고싶은것만 print 했다.
$ grep -n 'printf' fileio.c | grep -v 'chi' | grep -v 'cho'
11:             printf("FILE open Error\n");
25:             printf("FILE open Error\n");
```

```
## -Ev 옵션을 통해 여러번 grep 하는것을 파이프(|)로 작성할 수 있다.
$ grep -n 'printf' fileio.c | grep -Ev 'chi|cho'
11:             printf("FILE open Error\n");
25:             printf("FILE open Error\n");
```

```
grep -rl '#!.*python' | xargs sed -i '1s/python$/python3/'

// 하위 디렉토리의 모든 파일에서 1행이 #!로 시작해서 python을 끝나는 파일을 찾아서 python을 python3로 변경
#!/usr/bin/env python     # 원본
#!/usr/bin/env python3    # 변경
```

Gawk

```
df -ht ext4
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        40G   11G   27G  29% /
/dev/sde1       173M   95M   65M  60% /boot
/dev/sde2        19G  1.5G   16G   9% /mnt/debian
/dev/sda1       1.8T  9.7G  1.7T   1% /mnt/1st-bay
/dev/sdb1       1.8T  1.1G  1.7T   1% /mnt/2nd-bay
/dev/sdc1       1.8T   28K  1.7T   1% /mnt/3rd-bay
/dev/sdd1       1.8T   32K  1.7T   1% /mnt/4th-bay
```

```
# 읽어들인 NR 레코드(여기서는 Filesystem과 Size) 갯수가 2 이하일때는 awk에서 바로 출력하고
# 이후에는 출력결과를 sort를 거쳐서 출력
df -ht ext4 | awk 'NR<2{print $0;next}{print $0| "sort"}'
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        40G   11G   27G  29% /
/dev/sda1       1.8T  9.7G  1.7T   1% /mnt/1st-bay
/dev/sdb1       1.8T  1.1G  1.7T   1% /mnt/2nd-bay
/dev/sdc1       1.8T   28K  1.7T   1% /mnt/3rd-bay
/dev/sdd1       1.8T   32K  1.7T   1% /mnt/4th-bay
/dev/sde1       173M   95M   65M  60% /boot
/dev/sde2        19G  1.5G   16G   9% /mnt/debian
```

```
# 1q로 한줄만 sed에서 처리하고 종료 나머지는 sort가 처리
df -ht ext4 | { sed -u 1q; sort; }
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        40G   11G   27G  29% /
/dev/sda1       1.8T  9.7G  1.7T   1% /mnt/1st-bay
/dev/sdb1       1.8T  1.1G  1.7T   1% /mnt/2nd-bay
/dev/sdc1       1.8T   28K  1.7T   1% /mnt/3rd-bay
/dev/sdd1       1.8T   32K  1.7T   1% /mnt/4th-bay
/dev/sde1       173M   95M   65M  60% /boot
/dev/sde2        19G  1.5G   16G   9% /mnt/debian
```

wget

```
# -r / --recursive : 디렉토리 대상
# -np / --no-parent : 상위 제외
# -R / --reject : 해당 파일 제외
# -P / --directory-prefix=PREFIX : PREFIX/.. 에 파일 저장
wget -r -np -R "index.html*" -P <DIR> lfs/
https://ftp.osuosl.org/pub/lfs/lfs-packages/12.1/
```

Firmware Blob Kernel config

```
echo CONFIG_EXTRA_FIRMWARE='" '$({ cd /lib/firmware; echo amd-ucode/*; echo
amdgpu/re*; echo rtl_nic/*; echo regulatory*; })'"' >> .config
make oldconfig
```

Man page count

```
find /usr/share/man/ -type f -name '*.[0-9]' | wc -l
```