

--docdir

```
docdir () {
    local PATH_PREFIX_FOR_DOCDIR="/usr/share/doc"
    echo "${PATH_PREFIX_FOR_DOCDIR}/${(basename $(pwd))}"
}

export -f docdir
```

Host LFS chroot

6

```
:
auto-lfs s : chroot ,
auto-lfs c : chroot
```



LFS, LFS_PART /boot, /boot/efi, /home, /sources

```
#!/bin/bash
# LFS 12.1-systemd
# <7.2 >
# 가 /
# Begin ~/bin/lfs-env-config

VERSION=0.1.7
LFS=/mnt/lfs
LFS_PART=/dev/nvme0n1p8
LFS_BOOT=/dev/nvme0n1p6
LFS_EFI=/dev/nvme0n1p2
LFS_HOME=/dev/nvme0n1p7

#
if ! mountpoint -q $LFS; then mount $LFS_PART $LFS; fi
if ! mountpoint -q $LFS/boot; then mount $LFS_BOOT $LFS/boot; fi
if ! mountpoint -q $LFS/boot/efi; then mount $LFS_EFI $LFS/boot/efi; fi
if ! mountpoint -q $LFS/home; then mount $LFS_HOME $LFS/home; fi
if ! mountpoint -q $LFS/sources; then mount /mnt/nfs/sources $LFS/sources; fi

# /dev
if [ ! -d $LFS/dev ]; then
    echo "Make necessary directories"
    mkdir -pv $LFS/{dev,proc,sys,run}
```

```
else
    echo "All required directories have been verified."
fi

#          chroot          exit          가
case $1 in
    clean|c|-c)
        if mountpoint -q $LFS/dev/shm; then
            # Unmount Kernel Virtual File System used in previous LFS
            sessions.
            echo "LFS is now set to '$LFS'"
            echo "Now proceed with the unmount of <Kernel Virtual File
            System> for chroot"
            for kvfs in dev/shm dev/pts sys proc run dev ; do
                if mountpoint $LFS/$kvfs; then umount -q $LFS/$kvfs; fi
            done
            findmnt -D -R $LFS
            echo "Ok unmount unnecessary devices"
            echo "Good bye"
            exit 0
        else
            echo "Cleanup of Kernel Virtual File System is Complete."
            findmnt -D -R $LFS
            exit 0
        fi
    ;;
    start|s|-s)
        # Mount the Kernel Virtual File System to Enter LFS chroot
        environment.
        echo "Now proceed with the mount of <Kernel Virtual File System> for
        chroot"
        echo "LFS is now set to '$LFS'"
        if ! mountpoint -q $LFS/dev;      then mount --bind /dev $LFS/dev;
    fi
        if ! mountpoint -q $LFS/dev/pts; then mount -t devpts devpts -o
        gid=5,mode=0620 $LFS/dev/pts; fi
        if ! mountpoint -q $LFS/proc;    then mount -t proc proc $LFS/proc;
    fi
        if ! mountpoint -q $LFS/sys;     then mount -t sysfs sysfs $LFS/sys;
    fi
        if ! mountpoint -q $LFS/run;     then mount -t tmpfs tmpfs $LFS/run;
    fi
        if ! mountpoint -q $LFS/dev/shm; then mount -t tmpfs -o nosuid,nodev
        tmpfs $LFS/dev/shm;      fi
        if ! mountpoint -q $LFS/sources; then mount --rbind /mnt/nfs/sources
        $LFS/sources;          fi
        findmnt -D -R $LFS
        # Run 'chroot'
        chroot "$LFS" $(type -p env) -i \
            HOME=/root          \
            TERM="$TERM"        \

```

```

        PS1='<LFS> \u@\h [ \w ] \$ ' \
        PATH=/usr/bin:/usr/sbin \
        MAKEFLAGS="-j$(nproc)" \
        TESTSUITEFLAGS="-j$(nproc)" \
        NINJAJOBS="$(nproc)" \
        /bin/bash --login
    exit 0
;;
version|v|-v)
    echo "LFS chroot script $VERSION"
    exit 0
;;
help|h|-h)
    echo "LFS chroot script $VERSION"
    echo "Usage:"
    echo "clean    or c- Unmount Kernel Virtual File System"
    echo "start     or s- Mount and run 'chroot' command with necessary
variables"
    echo "version  or v- Print script version"
    echo "help     or h - This screen"
    exit 0
;;
*)
    #Guidance on the factors needed to execute script
    echo "sh ./lfs-env-config [ start | clean | help | version ]"
    exit 0
;;
esac
## End ~/bin/lfs-env-config

```

cd 가 .

```
alias <filename>='source <filename>'
```

:

가

1. URL Git 가
2. 가 .zip 가
3. 가
4. 가
- 5.

가

1. alias
2. wget, git, gawk

1. git clone source

```
user@host:[ ~ ] $ pre-stage
https://xorg.freedesktop.org/archive/individual/proto/xcb-proto-1.17.0.tar.x
z
xcb-proto-1.17.0.tar 100%
[=====>] 148.19K 302.76KB/s
          [Files: 1 Bytes: 148.19K [153.40KB/s] Redirects:
0 Todo: 0 Errors: 0 ]
user@host:[ /workbench/xcbe-proto-1.17.0 ] $ _
```

```
#!/bin/bash
```

```
Green='\033[0;32m'
```

```
BIGreen='\033[1;32m'
```

```
Red='\033[0;31m'
```

```
Yellow='\033[0;33m'
```

```
Color_Off='\033[0m'
```

```
if [ "$(type -t p4c)" != 'alias' ] ; then
    echo -e "p4c is not ${Yellow}alias registered${Color_Off}, so the
functionality is not available."
    source ${HOME}/.bash_aliases
    return 2>&/dev/null
fi
```

```
if [ -z "$1" ]; then echo -e "${Green}An appropriate factor (file name or
download link) is required.${Color_Off}"; return; fi
InputStringType=$(echo "$1" | awk 'BEGIN { FS = ":" } ; { print $1 }')
BuildBase="$LFS"/workbench
SourceBase="$LFS"/sources
TargetFile=$(basename "$1")
GitDirCheck=$(basename -s .git "$1")
GitCheck=${TargetFile: -3}
ZipCheck=${TargetFile: -3}
TargetDir=$(echo "$TargetFile" | sed -e 's/\.tar.*$//' -e 's/\.tgz*$//' -e
's/\.src*$//' -e 's/\.zip*$//' -e 's@^.*@/@')
NotFoundMsgl="The directory derived from the filename is
${BIGreen}$TargetDir${Color_Off}.\n\
However, it does not exist, so it will move to the ${Yellow}most recent
directory created${Color_Off}.\n\
Please check if it is correct."
```

```
if [ $(stat -c %u $BuildBase) -ne $(id -u) ] ; then echo -e "Check
${Red}$BuildBase${Color_Off} Ownership"; return; fi
```

```
function chg_dir() {
    if [ -d "$1" ]; then
        cd "$1"
    else
        cd $(ls -tcA -w1 | head -n1)
        echo -e "$NotFoundMsg1"
    fi
}

# Change directory "$BuildBase"
if [ "$PWD" != "$BuildBase" ]; then
    chg_dir "$BuildBase"
fi

## Handling Git
if [ "$GitCheck" == "git" ] ; then
    if [ -d "$GitDirCheck" ] ; then
        chg_dir "$GitDirCheck"
        return
    else
        pushd $SourceBase
        if [ -d $GitDirCheck ] ; then
            rm -rf $GitDirCheck
        fi
        git clone "$1"
        cp -R $GitDirCheck $BuildBase
        popd
#        echo "Git #1"
        chg_dir "$GitDirCheck"
        return
    fi
fi

case "$InputStringType" in
    https|http|ftp)
        wget --no-verbose -nc --directory-prefix="$SourceBase" "$1"
        ;;
    *)
        ;;
esac

# Prepare Package Build
if [ "$ZipCheck" == "zip" ] ; then
    mkdir -p "$TargetDir"
    chg_dir "$TargetDir"
    unzip -q "$SourceBase"/"$TargetFile"
    return
else
    if [ ! -d "$TargetDir" ]; then
        tar -xf "$SourceBase"/"$TargetFile" 2>/dev/null
    fi
fi
```

```
chg_dir "$TargetDir"
```

```
fi
```

```
unset InputStringType BuildBase SourceBase TargetFile GitDirCheck GitCheck  
ZipCheck TargetDir NotFoundMsg1
```

From:

<https://www.gamu.kr/dokuwiki/> -

Permanent link:

<https://www.gamu.kr/dokuwiki/linuxfromscratch/auto-lfs?rev=1768447714>

Last update: **2026/01/15 03:28**

