Własny Debian LiveUSB

(aktualizacja: 2021-06-08)

Live USB możemy bardzo prosto utworzyć instalując system z użyciem debootastrapa i uzupełniając go w chroocie o jądro, gruba i potrzebne nam narzędzia.

Artykuł ten prezentuje krok po kroku proces budowy własnego live usb. Zaprezentowane w nim zostały polecenia i konfiguracje m.in. związane z następującymi zagadnieniami:

- praca z obrazami urządzeń blokowych w plikach z użyciem losetup i innych narzędzi
- gpt; efi / uefi; grub;
- montowanie obrazu dysku z pliku (mount disk image from file);
- chroot; montowanie i odmontowywanie specjalnych systemów plików dla chroot; odmontowywanie systemów montowanych z opcją rbind (rbind umount);
- debootstrap;
- optymalizacja dla systemów na pamięciach flash: dane zmienne na tmpfs (ramdysk).

Nie jest to skrypt, który można uruchomić i czekać na gotowy produkt w postaci live USB. Jest to natomiast poradnik, który należy czytać i świadomie wykonywać kolejne podane w nim polecenia. Wymagane jest co najmniej ustawienie zmiennych DEV (wskazującej na urządzenie blokowe na którym prowadzimy instalację) oraz MNT wskazujące na punkt montowania tworzonego rootfs'a. Część prezentowanych ustawień jest opcjonalnych (np. konfiguracja apt'a) i odzwierciedla preferencje autora.

Obraz w pliku

Utworzenie obrazu dysku w pliku, a następnie przekopiowanie go przy pomocy dd na nośnik, jest opcjonalne, ale niekiedy może znacząco przyspieszyć proces instalacji.

```
dd if=/dev/zero of=usb.img bs=1M count=3699
DEV=`losetup --partscan --find --show usb.img`
```

Powyższe wywołanie `losetup` może być używane także do tworzenia urządzeń związanych z już istniejącym plikiem obrazu (a nie tylko nowo utworzonego pliku przez dd) – w takiej sytuacji od razu pojawią się też urządzenia związane z istniejącymi w obrazie partycjami.

Polecenie `losetup` może być też zastąpione odpowiednimi wywołaniami polecenia `kpartx`, np:

```
kpartx -va $IMAGE; DEV=`kpartx -l $IMAGE | awk '/^loop[0-9]/ { print $5; exit; }'`
```

Możliwe jest aby obraz zawierał też wolumeny LVM w takim wypadku po zmapowaniu obrazu z użyciem `losetup` konieczne może być wywołanie `lvscan` aby wyświetlić dostępne wolumeny `lvchange -ay WOLUMEN` aby dokonać ich aktywacji a przed usunięciem mapowania przez `losetup -d` ich dezaktywacja przy pomocy `lvchange -an WOLUMEN`.

Przygotowanie tablicy partycji

```
# tworzymy tablicę partycji typu gpt
# zasadniczo wystarczył by zwykły MBR z partycją EFI, ale zrobimy hybrydę GPT + MBR
parted $DEV "mklabel gpt"

# tworzy partycję która posłuży do wgrania gruba (musi mieć co najmniej 128 kiB)
# należy zaznaczyć iż nie jest to partycja /boot
# jest to surowe (bez filesystemu) miejsce na dysku gdzie zostanie wgrany
# fragment gruba normalnie wgrywany do MBR
parted $DEV "mkpart grub 0 2MB";

# ustawiamy dla tej partycji flagę "GRUB BIOS partition"
parted $DEV "set 1 bios_grub on";

parted $DEV "mkpart efi 2MB 200MB";
parted $DEV "set 2 boot on";

# resztę dysku możemy podzielić wg uznania,
# tutaj robimy jedną dużą partycję o nazwie rootfs
parted $DEV "mkpart rootfs 200MB 100%";

# update kernel table — in most case we don't need this
# kpartx -u $DEV

# przy pomocy gdisk utworzenie hybrid MBR
gdisk $DEV
# r ->
#      o
#      h ->
#           1 2 3
#           n 83 n
#           ef y
#           83 n
#      o
#      w ->
```

```
#           y

if [ -e ${DEV}1 ]; then
        DEVp=${DEV}
elif [ -e ${DEV}p1 ]; then
        DEVp=${DEV}p
else
        print "ERROR cant find device for partitions on ${DEV}"
fi

mkfs.vfat -F 32 ${DEVp}2
mkfs.xfs ${DEVp}3
```

Instalacja systemu

```
# zamontowanie systemu plików
mount ${DEVp}3 ${MNT}

# ustawiamy adres mirrora i instalowaną gałąź Debiana
# (chyba że był wcześniej były ustawione)
MIRROR=${MIRROR:="http://ftp.icm.edu.pl/pub/Linux/debian/"}
SUITE="stable"

# instalujemy system debootstrapem
debootstrap ${SUITE} ${MNT} ${MIRROR}
```

Chroot

```
# zamontowanie /proc, /sys i /dev na potrzeby chroot'a
mount -t proc  proc  $MNT/proc
mount -t sysfs sysfs $MNT/sys
mount -o rbind /dev  $MNT/dev
mount -o rbind /run  $MNT/run/

# ustawienie zmiennych środowiskowych
export DEV DEVp MIRROR SUITE
export LC_ALL="C.UTF-8"

# chroot
chroot $MNT
```

Uwaga: wszystkie następne kroki wykonywane są wewnątrz chroot'a

Podstawowa konfiguracja systemu

```
# używamy UTF-8
echo 'LANG="C.UTF-8"' > /etc/default/locale;
echo 'set -a; . /etc/default/locale; set +a' > /etc/profile.d/locale.sh
echo 'perl -e exit 2>&1 | grep "Setting locale failed" >/dev/null 2>&1 && export LC_ALL=C.UTF-8' >> /etc/profile.d/locale.sh

# ustalamy UUID rootfs obrazu
export `blkid -o export ${DEVp}3 | grep ^UUID=`

# w /etc/fstab ustawiamy rootfs na read-only, a /tmp na tmpfs
# /var/run i /var/lock są współcześnie linkami do /run na tmpfs
cat << EOF > /etc/fstab
/dev/disk/by-uuid/$UUID  /     xfs    ro        0  0
tmpfs                    /tmp  tmpfs  defaults  0  0
EOF

# katalogi często pisane na ramdysku
# /tmp będzie punktem montowania więc go czyścimy
rm -fr /var/tmp;   ln -s /run/tmp   /var/tmp
rm -fr /var/log;   ln -s /run/log   /var/log

cat > /etc/tmpfiles.d/on_tmpfs.conf <<EOF
d  /run/tmp   1777 root root -
L+ /var/tmp   -    -    -    -  /run/tmp
d  /run/log   0755 root root -
L+ /var/log   -    -    -    -  /run/log
EOF

# konfigurujemy zapis journala na ramfs
sed -e 's@^.*Storage=.*@Storage=volatile@' -i /etc/systemd/journald.conf

# linkujemy /proc/mounts do /etc/mtab
ln -s /proc/mounts /etc/mtab
```

Chroot - dodajemy ramfs'y

```
mount /tmp
```

```
# /run jest montowany wcześniej jako rbind, ale może nie zawierać log i tmp
# mount -t tmpfs tmpfs /run
mkdir -p /run/{lock,log,tmp}
```

Podstawowa konfiguracja APT

```
cat << EOF > /etc/apt/apt.conf.d/03misc
# nie trzyma polecanych i sugerowanych jak zalezności
APT::AutoRemove::RecommendsImportant "false";
APT::AutoRemove::SuggestsImportant "false";

# nie instaluje automatycznie polecanych i sugerowanych
APT::Install-Recommends "false";
APT::Install-Suggests "false";

# wyłączenie wyszukiwania przyrostowego w aptitude
aptitude::UI::Incremental-Search "false";

# wyłączenie pobierania aktualizacji listy pakietów jako pdiff
# (polecam jeżeli stosunkowo rzadko aktualizujemy listy pakietów np. na testingu)
Acquire::PDiffs "false";
EOF

echo "deb $MIRROR $SUITE main" > /etc/apt/sources.list
apt-get update

apt-get install aptitude
```

Instalacja przydatnych narzędzi i programów

```
SYSBASE="aptitude bsdmainutils cron less rsync vim xxd
        procps psmisc lsof htop iotop strace
        schroot busybox file screen tmux sudo bash-completion
        konwert gpg gpgv gpgsm bzip2 xz-utils p7zip-full
        python3 python3-pip whiptail gawk bc man-db manpages
        pass pwgen  console-setup gpm"

HWTOOLS="sysstat lm-sensors hddtemp
        pciutils usbutils lsscsi dmidecode
        hdparm sdparm smartmontools
        picocom jpnevulator
        ipmitool pcmciautils"
FSTOOLS="lvm2 mdadm dmsetup multipath-tools cryptsetup-bin
        parted fdisk gdisk kpartx testdisk
        e2fsprogs xfsprogs btrfs-progs dosfstools
        nfs-kernel-server"

NETBASE="iproute2 nftables iputils-ping isc-dhcp-client
        net-tools iputils-tracepath dnsutils mtr-tiny
        ethtool vlan bridge-utils iptables arptables ebtables
        netcat-openbsd ncat socat telnet
        wget curl httrack ntpdate debootstrap
        openssh-client openssh-server sshfs sshguard"
NETDIAG="inetutils-traceroute traceroute paris-traceroute tcptraceroute
        tcpdump nmap
        sipcalc ipv6calc
        arping arp-scan ndisc6
        vnstat iftop netdiag tcpflow ngrep
        dnstop dnstracer dnswalk"
NETWIFI=" wireless-tools wpasupplicant hostapd rfkill"

DEVUTILS="git subversion patch"

X11TOOL="xorg openbox firefox-esr ca-certificates xpdf xterm wireshark"

aptitude install $SYSBASE $HWTOOLS $FSTOOLS $NETBASE $NETDIAG $NETWIFI $DEVUTILS $X11TOOL
aptitude purge   vim-tiny mawk nano tasksel tasksel-data debconf-i18n gdbm-l10n
aptitude clean

systemctl disable vnstat.service
systemctl disable hostapd.service
```

Grub i jądro

```
# włączamy non-free
echo "deb $MIRROR $SUITE main contrib non-free" > /etc/apt/sources.list
aptitude update

# instalacja pakietów gruba, jądra oraz narzędzi efi
aptitude install grub2 grub-efi-amd64-bin efivar efibootmgr \
                 linux-image-amd64 firmware-linux \
                 firmware-realtek ntfs-3g
```

```
aptitude clean

# plik konfiguracyjny GRUBa
UUID=`grub-probe -d ${DEVp}3 -t fs_uuid`
cat << EOF > /boot/grub/grub.cfg
# korzystanie z konsoli na porcie szeregowym
#serial --speed=115200 --unit=1 --word=8 --parity=no --stop=1
#terminal_input serial console
#terminal_output serial console
# za to na którym RS jest grub odpowiada --unit w serial
#
# korespondujące opcje uruchamiania dla Linuxa:
#   module    /boot/vmlinuz [...] console=tty0 console=ttyS1,115200n8
# korespondujące opcje uruchamiania dla XENa:
#   multiboot /boot/xen.gz  [...] com2=115200,8n1 vga=mode-0x0319 console=com2,vga
#   module    /boot/vmlinuz [...] console=tty0 console=hvc0
#
# w opcjach kernela szczególną rolę odgrywa ostatni podany terminal (ostatnia opcja console)
# jest on związany z /dev/console i na nim pokazywany jest init, ponadto należy uruchamiać getty
# na odpowiednim urządzeniu poprzez wpis w inittab np.:
#   T0:23:respawn:/sbin/getty -L /dev/ttyS1 115200 vt100
# lub (dla wariantu XENowego, UWAGA: w przykładzie tylko runlevel 4 i 5):
#   T1:45:respawn:/sbin/getty 38400 hvc0

# timeout w sekundach dla domyślnej pozycji menu
set timeout=60
set default="0"

# ładowanie modułów - typ tablicy partycji
insmod part_gpt
insmod part_msdos

# ładowanie modułów - RAID
insmod mdraid1x # mdraid metadata 1.x
insmod mdraid09 # mdraid metadata 0.9
#insmod raid5rec # faulty RAID4/5
#insmod raid6rec # faulty RAID6

# ładowanie modułów - LVM
insmod lvm

# ładowanie modułów - File System
insmod xfs

# ustawienie root'a
search --no-floppy --fs-uuid --set=root $UUID


# ustawienia wizualne
if [ x\$feature_default_font_path = xy ] ; then
    font=unicode
else
    font="/usr/share/grub/unicode.pf2"
fi

if loadfont \$font ; then
  if [ x\$feature_all_video_module = xy ]; then
    insmod all_video
  else
    insmod efi_gop
    insmod efi_uga
    insmod ieee1275_fb
    insmod vbe
    insmod vga
    insmod video_bochs
    insmod video_cirrus
  fi
  insmod gfxterm
  insmod gettext
fi

set menu_color_normal=white/red
set menu_color_highlight=yellow/black
set pager=1
terminal_output gfxterm


# pozycje menu
menuentry 'Debian GNU/Linux USB-LIVE by rrp@opcode.eu.org' {
  echo    'Loading Linux ...'
  linux   /vmlinuz root=UUID=$UUID rootdelay=5 net.ifnames=0 ro
  echo    'Loading initial ramdisk ...'
  initrd  /initrd.img
}

menuentry 'Debian GNU/Linux USB-LIVE by rrp@opcode.eu.org init=/bin/bash' {
```

```
  echo    'Loading Linux ...'
  linux   /vmlinuz root=UUID=$UUID rootdelay=5 net.ifnames=0 ro init=/bin/bash
  echo    'Loading initial ramdisk ...'
  initrd  /initrd.img
}
EOF

# instalacja GRUBa dla UEFI
mkdir /boot/efi
mount ${DEVp}2 /boot/efi
grub-install --efi-directory=/boot/efi --target=x86_64-efi
mkdir /boot/efi/EFI/BOOT
cp /boot/efi/EFI/debian/grubx64.efi /boot/efi/EFI/BOOT/bootx64.efi

# instalacja GRUBa w MBR
grub-install --boot-directory=/boot/ --target=i386-pc $DEV
```

Konfiguracja końcowa

```
# automatyczne podnoszenie sieci
echo "allow-hotplug eth0" > /etc/network/interfaces.d/eth0
echo "iface eth0 inet dhcp" >> /etc/network/interfaces.d/eth0
echo "nameserver 8.8.8.8" > /etc/resolv.conf

# w /etc/rc.local podnosimy wszystkie interfejsy
cat << EOF > /etc/rc.local
#!/bin/bash

for i in \`ip l | awk '/^[0-9]/ {print \$2}' | tr -d :\`; do
        ifconfig \$i up
        ip link set dev \$i
done
EOF

cat << EOF > /etc/systemd/system/rc-local.service
[Unit]
Description=/etc/rc.local
ConditionPathExists=/etc/rc.local

[Service]
Type=forking
ExecStart=/etc/rc.local start
TimeoutSec=0
StandardOutput=tty
RemainAfterExit=yes
SysVStartPriority=99

[Install]
WantedBy=multi-user.target
EOF

chmod +x /etc/rc.local
systemctl enable rc-local

# ssh pozwala na logowanie na root'a i nie jest automatycznie uruchamine
sed -e 's|^[# ]*PermitRootLogin.*$|PermitRootLogin yes|' -i /etc/ssh/sshd_config
systemctl  disable  ssh.service

# różne ustawienia
systemctl  mask  hwclock-save.service systemd-random-seed.service udev-finish.service \
                 systemd-update-utmp-runlevel.service systemd-update-utmp.service
rm /usr/lib/modules-load.d/ipmievd.conf
echo "liveUSB" > /etc/hostname

# hasło root'a
passwd

# konfiguracja użytkownika - skrypty do montowania i odmontowywania chroot'ów
# oraz fałszywa historia linii poleceń (z tym co może nam się przydać w przyszłości)
cat << EOF > /root/prepareChroot.sh
#!/bin/bash

if [ \$# -ne 2 ]; then
        echo USAGE: \$0 device mountpoint
        exit
fi

MNTDEV=\$1
MNT=\$2

mount \$MNTDEV \$MNT
mount -t proc  proc  \$MNT/proc
mount -t sysfs sysfs \$MNT/sys
mount -o rbind /dev  \$MNT/dev
mount -o rbind /run  \$MNT/run/
```

```
chroot \$MNT
EOF
chmod +x /root/prepareChroot.sh

cat << EOF > /root/umountChroot.sh
if [ \$# -ne 1 ]; then
        echo USAGE: \$0 mountpoint
        exit
fi

MNT=\$1

umount \$MNT/proc
umount \$MNT/sys
mount --make-rslave \$MNT/dev && umount -R \$MNT/dev
mount --make-rslave \$MNT/run && umount -R \$MNT/run
umount \$MNT
EOF
chmod +x /root/umountChroot.sh

cat << EOF > /root/runSSH.sh
#!/bin/bash

if [ \$# -ne 1 ]; then echo "USAGE \$0 client_ip"; exit; fi

nft add table inet filter; nft flush table inet filter;
nft add chain inet filter INPUT '{ type filter hook input priority 0; }';
if echo "\$1" | grep : >/dev/null; then
        nft add rule  inet filter INPUT ip6 saddr "\$1" accept
else
        nft add rule  inet filter INPUT ip saddr "\$1" accept
fi
nft add rule  inet filter INPUT tcp dport 22 reject

systemctl start ssh.service
EOF
chmod +x /root/runSSH.sh

cat << EOF > /root/.bash_history
HISTFILE=/dev/null; mount -o remount,rw /
/root/runSSH.sh 192.168.13.0/24
/root/prepareChroot.sh /dev/sdb1 /mnt/1
chroot /mnt/1
/root/umountChroot.sh /mnt/1
EOF
ln -s /root/.bash_history /.bash_history

# samobójstwo bash'a ... zapobiega zapisaniu historii linii poleceń
# skutkuje też wyjściem z chroot'a
kill -9 $BASHPID
```

Odmontowanie chroot'a

Uwaga: wszystkie następne kroki wykonywane poza chroot-em

```
umount $MNT/boot/efi/
umount $MNT/run
umount $MNT/tmp
umount $MNT/proc
umount $MNT/sys
mount --make-rslave $MNT/dev
umount -R $MNT/dev
mount --make-rslave $MNT/run
umount -R $MNT/run
umount $MNT
losetup -d $DEV
```

Nagranie obrazu na USB

Jeżeli obraz tworzony był w pliku należy skopiować go na nośnik USB np. przy użyciu komendy `dd`:

```
dd status=progress oflag=sync bs=1M if=usb.img of=ŚCIEŻKA_DO_URZĄDZENIA_USB
```

gdzie `ŚCIEŻKA_DO_URZĄDZENIA_USB` jest ścieżką do urządzania (nie partycji) związanego z nośnikiem USB (np. `/dev/sdf`).

Licencja

ARE PERMITTED under the terms of the MIT license.

The MIT License:

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.