13. BusyBox

Sestavení Linuxového systému
Cíl cvičení
Zadání úlohy
Evaluace
Nápověda
Tvorba souborového systému
Bootování jádra s naším filesystémem v emulátoru Qemu
Možná vylepšení
Tipy a triky
Reference

Sestavení Linuxového systému

Cíl cvičení

Na předchozích cvičeních jsme se zabývali jednotlivými aspekty operačních systémů a jejich používání a programování. V této dobrovolné úloze si zkusíte sestavit kompletní operační systém z jednotlivých komponent. Bude se jednat o operační systém s jádrem Linux a jako uživatelský prostor budeme používat projekt BusyBox. Tato kombinace komponent je často používána ve vestavěných (embedded) aplikacích, jako jsou například síťové routery, chytré domácí spotřebiče apod.

Co je Linuxové jádro snad už trochu tušíte z přednášek. BusyBox je sada miniaturních Unixových programů jako je například příkazový interpret (shell), editor, příkazy jako `ls`, `find`, `cat`, atd. Pomocí projektu BusyBox lze vytvořit funkční unixové uživatelské prostředí o celkové velikosti pod 1 MiB.

Zadání úlohy

> Tato úloha není povinná.

Vytvořte shellový skript, který automaticky vytvoří a v emulátoru Qemu nabootuje funkční operační systém. Skript by měl vykonat následující operace:

- Stáhne zdrojové kódy projektu BusyBox (buď z archivu nebo z repozitáře verzovacího systému git).
- Nakonfiguruje a zkompiluje BusyBox (stačí výchozí konfigurace pomocí `make defconfig`, kompilace pomocí `make`).
- Stáhne zdrojové kódy jádra Linux. Buď jako archiv tar nebo naklonuje repozitář gitu.
- Nakonfiguruje (opět stačí `make defconfig`) a zkompiluje jádro. Doporučujeme kompilovat paralelně na všech dostupných procesorech pomocí `make -j$(nproc)`.
- Ze zkompilovaného BusyBoxu vytvoří souborový systém tak, aby Linuxové jádro mohlo nabootovat. Viz nápovědu níže.
- Zkompilované jádro a vytvořený souborový systém nabootuje v emulátoru Qemu. (Pozor: V předchozích cvičeních jsme používali emulátory 32bitového systému. Zde budete pravděpodobně kompilovat 64bitový systém, takže použijte i jeho emulátor `qemu-system-x86_64`).

Evaluace

Úloha se nekontroluje plně automaticky - je proto nutná kontrola cvičícím! Buď osobně na cvičení nebo v BRUTE. V BRUTE to stačí pokud tam váš skript funguje, t.j. ve výstupu AE je vidět hláška `Please press Enter to activate this console`.

Nápověda

Tvorba souborového systému

Základem pro váš souborový systém bude BusyBox nainstalovaný příkazem

```
make install
```

Tím se vytvoří základní adresářová struktura, ale k nabootování systému to stačit nebude. Bude potřeba vyřešit následující nedostatky:

1. Doplnit sdílené knihovny, které potřebuje BusyBox ke svému běhu.
2. Vytvořit adresář `/dev` s odkazy na nejnutnější zařízení.

Jaké jsou potřeba knihovny pro běh daného programu lze zjistit příkazem `ldd`. V našem případě tedy spustíme

```
ldd _install/bin/busybox
```

Všechny zmiňované knihovny (s výjimkou `linux-vdso.so.1`) je potřeba mít v cílovém souborovém systému v místě, kde je najde dynamický linker (`/lib64/ld-linux-x86-64.so.2`). ěTeoreticky byste si i tyto knihovny mohli zkompilovat sami, ale vzhledem k tomu, že kompilace knihovny jazyka C (`libc`) není triviální, použijte už zkompilovanou knihovnu z vašeho systému.

Dostupnost všech potřebných knihoven můžete otestovat příkazem `chroot`. Pokud se rozhodnete svůj souborový systém vytvářet v adresáři `_install`, spusťte tedy

```
chroot _install /bin/sh
```

Ten spustí příkaz `/bin/sh` s kořenovým adresářem nastaveným na `_install`. Shell tedy neuvidí žádné jiné soubory než ty, co jsou v adresáři `_install` a pokud tam budou chybět potřebné knihovny, nepůjde ani spustit.

Nejjednodušší možnost jak nabootovat do právě vytvořeného uživatelského prostředí je uložit ho ve formátu pro Linuxový startovací RAM-disk a nabootovat Linuxové jádro s tímto RAM-diskem. Aby vše fungovalo jak má, kromě souborů v adresáři `_install` musí RAM-disk obsahovat i několik položek v adresáři `/dev` pro přístup k virtuálním terminálům. V závislosti na vašem oprávnění můžete RAM-disk vytvořit jedním z následujících způsobů:

- Bez rootovských práv (což je i případ BRUTE) můžete RAM-disk vytvořit pomocí nástroje `gen_init_cpio`. Pokud program nemáte na svém počítači, měl by se vám zkompilovat pří kompilaci Linuxového jádra do adresáře `usr` (pozor, ne `/usr`).

```
(
cat <<EOF
dir /dev 755 0 0
nod /dev/tty0 644 0 0 c 4 0
nod /dev/tty1 644 0 0 c 4 1
nod /dev/tty2 644 0 0 c 4 2
nod /dev/tty3 644 0 0 c 4 3
nod /dev/tty4 644 0 0 c 4 4
slink /init bin/busybox 700 0 0
dir /proc 755 0 0
dir /sys 755 0 0
EOF

find _install -mindepth 1 -type d -printf "dir /%P %m 0 0\n"
find _install -type f -printf "file /%P %p %m 0 0\n"
find _install -type l -printf "slink /%P %l %m 0 0\n"
```

```
) > filelist

gen_init_cpio filelist | gzip > ramdisk
```

Výše uvedené příkazy fungují následovně: Nejprve vytvoříme seznam souborů (`filelist`), které má ramdisk obsahovat. Nástroj `gen_init_cpio` pak podle toho seznamu vytvoří obraz ramdisku, který "zazipujeme" příkazem `gzip` a uložíme do souboru `ramdisk` .

- Pokud máte *rootovská* práva, můžete odkazy na zařízení zkopírovat z vašeho systému a RAM-disk pak vytvořit následovně:

```
mkdir _install/{dev,etc,proc,sys}
sudo cp -a /dev/tty? _install/dev
ln -s bin/busybox _install/init
(cd _install; find . | cpio -o -H newc | gzip) > ramdisk
```

Bootování jádra s naším filesystémem v emulátoru Qemu

Na 64-bitovém systému spustíme emulátor Qemu s parametry `-kernel` a `-initrd` . Jako jádro zvolíme výsledek naší kompilace jádra, typicky soubor `arch/x86/boot/bzImage` . Spuštění emulátoru pak může vypadat např. následovně:

```
qemu-system-x86_64 -kernel linux-stable/arch/x86/boot/bzImage -initrd busybox
```

Pokud vše proběhlo správně, zobrazila se hláška

```
Please press Enter to activate this console.
```

a po stisku `Enteru` můžete začít pracovat ve vašem právě vytvořeném systému.

Možná vylepšení

Dále můžete provést drobná (či větší) vylepšení svého nového systému, která vám mohou zjednodušit další práci.

- Můžete připojit souborový systém `/proc` , aby fungovaly příkazy jako např. `ps` (výpis běžících procesů). Příkaz spusťte v emulátoru, ne na vaší pracovní stanici.

```
mount -t proc none /proc
```

- V RAM-disku můžete vytvořit soubor `/etc/init.d/rcS`, který bude obsahovat příkazy, které budou spuštěny při bootu systému.

```
mkdir -p _install/etc/init.d cat <<EOF >_install/etc/init.d/rcS
#!/bin/sh
mount -t proc none /proc
echo Nazdar!!!!
EOF
chmod +x _install/etc/init.d/rcS # nastavení spustitelnosti
```

  Nyní musíte znovu vytvořit RAM-disk a nabootovat.

- Zachytávání zpráv jádra spuštěného v emulátoru QEMU do souboru. Zprávy jádra je možné přesměrovat na virtuální sériový port a odtamtud pak například na standardní výstup:

```
qemu -serial stdio ...
```

  a jádru předáme parametr `console=ttyS0`

```
qemu -serial stdio -append console=ttyS0 ...
```

- Pokud chcete z vašeho systému komunikovat po síti, připojte ho na vnější síť pomocí NAT na uživatelské úrovni:

```
qemu -net nic,vlan=0,model=ne2k_pci -net user,vlan=0 ...
```

Tipy a triky

- Pokud je Qemu spouštěný přes vzdálené připojení (např. server postel), je potřeba pro zobrazení emulované obrazovky spouštěného stroje buď provést protunelování X protokolu (`ssh -X`) nebo používat Qemu s emulací obrazovky v textovém režimu `qemu -curses`. Další možnost je emulovat HW bez grafické karty `qemu -nographic` a nastavit testovaný systém tak, aby systémová konzole směřovala na sériový port (`-append console=ttyS0`).

Reference

- Podobná úloha v předmětu OSP
- ramfs, rootfs and initramfs
- Early userspace support
- Inside the Linux boot process
- The Linux Bootdisk HOWTO