2023-06-10

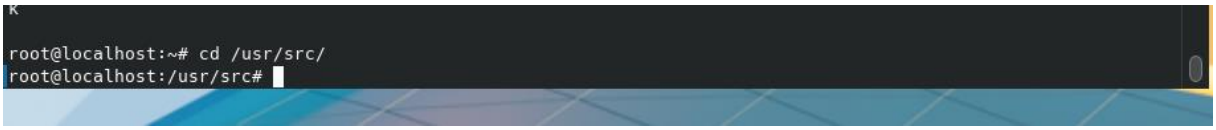
Kompilacja jądra systemu Linux

„Stara” i „Nowa” metoda

Paweł Paluch
304611

Stara metoda

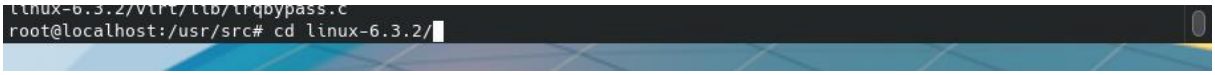
1. `cd /usr/src/` - zmienia bieżący katalog na `/usr/src/`.



2. `wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.3.2.tar.xz` - pobiera plik `linux-6.3.2.tar.xz` z podanego adresu URL.

3. `tar -xvpf linux-6.3.2.tar.xz` - rozpakowuje plik `linux-6.3.2.tar.xz`.

4. `cd linux-6.3.2/` – zmienia bieżący katalog na `linux-6.3.2/`.



5. `zcat /proc/config.gz > .config` - kopiuje zawartość pliku `/proc/config.gz` do pliku `.config`.

6. `make localmodconfig` - tworzy plik konfiguracyjny dla bieżącej konfiguracji systemu.

7. `make -j6 bzImage` - buduje obraz jądra `bzImage`, używając 6 wątków.

8. `make -j6 modules` - buduje moduły jądra, używając 6 wątków.

```
Kernel: arch/x86/boot/bzImage is ready  (#2)
root@localhost:/usr/src/linux-6.3.2# make -j6 modules
```

9. `make -j6 modules_install` - instaluje moduły jądra, używając 6 wątków.

```
  LD [M]  net/rfkill/rfkill.ko
root@localhost:/usr/src/linux-6.3.2# make -j6 modules_install
```

10. `cp arch/x86/boot/bzImage /boot/vmlinuz-old-6.3.2-smp` - kopiuje obraz jądra do katalogu `/boot/`

```
  INSTALL /lib/modules/6.3.2-smp/kernel/net/rfkill/rfkill.ko
  DEPMOD  /lib/modules/6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp arch/x86/boot/bzImage /boot/vmlinuz-old-6.3.2-smp
```

11. `cp System.map /boot/System.map-old-6.3.2-smp` - kopiuje mapę symboli do katalogu `/boot/`

```
root@localhost:/usr/src/linux-6.3.2# cp arch/x86/boot/bzImage /boot/vmlinuz-old-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp System.map /boot/System.map-old-6.3.2-smp
```

12. `cp .config /boot/config-old-6.3.2-smp` - kopiuje plik konfiguracyjny do katalogu `/boot/`

```
root@localhost:/usr/src/linux-6.3.2# cp System.map /boot/System.map-old-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp .config /boot/config-old-6.3.2-smp
```

13. `cd /boot/` - zmienia bieżący katalog na `/boot/`

```
root@localhost:/usr/src/linux-6.3.2# cp .config /boot/config-old-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cd /boot/
root@localhost:/boot#
```

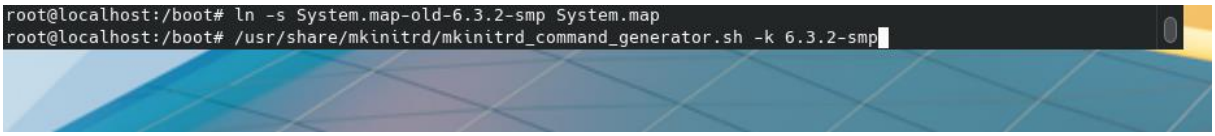
14. `rm System.map` - usuwa mapę symboli.

```
root@localhost:/usr/src/linux-6.3.2# cd /boot/
root@localhost:/boot# rm System.map
root@localhost:/boot#
```

15. `ln -s System.map-old-6.3.2-smp System.map` - tworzy dowiązanie symboliczne do nowej mapy symboli.

```
root@localhost:/boot# rm System.map
root@localhost:/boot# ln -s System.map-old-6.3.2-smp System.map
```

16. `/usr/share/mkinitrd/mkinitrd_command_generator.sh -k 6.3.2-smp` - generuje polecenie do utworzenia initrd dla wersji jądra `6.3.2-smp`

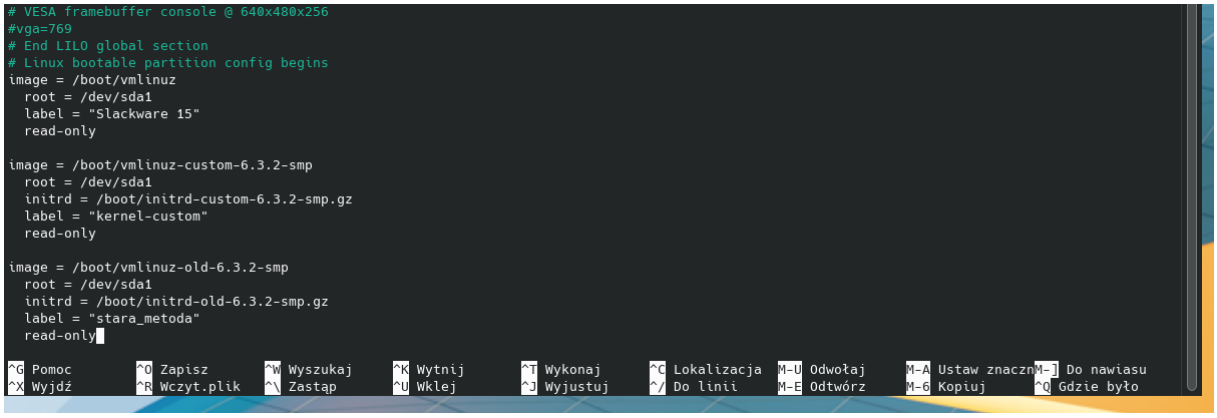


17. `mkinitrd -c -k 6.3.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-old-6.3.2-smp.gz` - tworzy initrd dla wersji jądra `6.3.2-smp`.

18. `nano /etc/lilo.conf` - otwiera plik konfiguracyjny bootloadera LILO w edytorze tekstu nano.

19. Dodanie wpisu do lilo.conf:

```
image = /boot/vmlinuz-old-6.3.2-smp
  root = /dev/sda1
  initrd = /boot/initrd-old-6.3.2-smp.gz
  label = "stara_metoda"
  read-only
```



Ten wpis w pliku konfiguracyjnym bootloadera LILO określa opcję uruchamiania systemu z jądrem znajdującym się w pliku `/boot/vmlinuz-old-6.3.2-smp`, z partycją root `/dev/sda1`, initrd `/boot/initrd-old-6.3.2-smp.gz`, etykietą `stara_metoda` i opcją `read-only`, która montuje partycję root tylko do odczytu podczas uruchamiania systemu.

20. `lilo` i `reboot` - polecenie `lilo` aktualizuje bootloader LILO zgodnie z wpisami w pliku konfiguracyjnym `/etc/lilo.conf`. Polecenie `reboot` restartuje system. Po restarcie systemu można wybrać opcję uruchamiania systemu z nowym jądrem, korzystając z menu bootloadera LILO.

21. `uname -r` - polecenie `uname` z opcją `-r` wyświetla wersję aktualnie uruchomionego jądra systemu.

Nowa metoda

1. `cd /usr/src/` - zmienia bieżący katalog na `/usr/src/`.
2. `rm -R linux-6.3.2` - usuwa katalog `linux-6.3.2` wraz z jego zawartością.
3. `tar -xvpf linux-6.3.2.tar.xz` - rozpakowuje plik `linux-6.3.2.tar.xz`.
4. `cd linux-6.3.2/` - zmienia bieżący katalog na `linux-6.3.2/`.
5. `cp /boot/config .config` - kopiuje plik konfiguracyjny z katalogu `/boot/` do bieżącego katalogu.

```
root@localhost:/usr/src# cp /boot/config .config
root@localhost:/usr/src#
```

6. `./scripts/kconfig/streamline_config.pl > config_strip` - uruchamia skrypt `streamline_config.pl`, który usuwa niepotrzebne opcje z pliku konfiguracyjnego i zapisuje wynik do pliku `config_strip`.

```
root@localhost:/usr/src# cd linux-6.3.2
root@localhost:/usr/src/linux-6.3.2# ./scripts/kconfig/streamline_config.pl > config_strip
using config: '/proc/config.gz'
root@localhost:/usr/src/linux-6.3.2#
```

7. `mv .config config.bak` - zmienia nazwę pliku `.config` na `config.bak`.

```
root@localhost:/usr/src/linux-6.3.2# cp /boot/config .config
root@localhost:/usr/src/linux-6.3.2# mv .config config.bak
root@localhost:/usr/src/linux-6.3.2# mv config_strip .config
root@localhost:/usr/src/linux-6.3.2# make oldconfig
```

8. `mv config_strip .config` - zmienia nazwę pliku `config_strip` na `.config`

```
root@localhost:/usr/src/linux-6.3.2# cp /boot/config .config
root@localhost:/usr/src/linux-6.3.2# mv .config config.bak
root@localhost:/usr/src/linux-6.3.2# mv config_strip .config
root@localhost:/usr/src/linux-6.3.2# make oldconfig
```

9. `make oldconfig` - aktualizuje plik konfiguracyjny dla nowej wersji jądra

```
root@localhost:/usr/src/linux-6.3.2# cp /boot/config .config
root@localhost:/usr/src/linux-6.3.2# mv .config config.bak
root@localhost:/usr/src/linux-6.3.2# mv config_strip .config
root@localhost:/usr/src/linux-6.3.2# make oldconfig
```

10. `make -j6 bzImage` - buduje obraz jądra `bzImage`, używając 6 wątków

```
root@localhost:/usr/src/linux-6.3.2# make oldconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  HOSTCC  scripts/kconfig/menu.o
  HOSTCC  scripts/kconfig/parser.tab.o
  HOSTCC  scripts/kconfig/preprocess.o
  HOSTCC  scripts/kconfig/symbol.o
  HOSTCC  scripts/kconfig/util.o
  HOSTLD  scripts/kconfig/conf
#
# configuration written to .config
#
root@localhost:/usr/src/linux-6.3.2# make -j6 bzImage
```

11. `make -j6 modules` - buduje moduły jądra, używając 6 wątków.

```
  BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready  (#1)
root@localhost:/usr/src/linux-6.3.2#
```

```
  BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready  (#1)
root@localhost:/usr/src/linux-6.3.2# make -j6 modules
```

12. `make -j6 modules_install` - instaluje moduły jądra, używając 6 wątków

```
  LD [M]  net/llc/llc.ko
  LD [M]  net/8021q/8021q.ko
  LD [M]  net/rfkill/rfkill.ko
root@localhost:/usr/src/linux-6.3.2# make -j6 modules_install
```

13. `cp arch/x86/boot/bzImage /boot/vmlinuz-new-6.3.2-smp` - kopiuje obraz jądra do katalogu `/boot/`

```
root@localhost:/usr/src/linux-6.3.2# cp arch/x86/boot/bzImage /boot/vmlinuz-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp System.map /boot/System.map-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp .config /boot/vmlinuz-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cd /boot/
root@localhost:/boot#
```

14. `cp System.map /boot/System.map-new-6.3.2-smp` - kopiuje mapę symboli do katalogu `/boot/`

```
root@localhost:/usr/src/linux-6.3.2# cp arch/x86/boot/bzImage /boot/vmlinuz-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp System.map /boot/System.map-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp .config /boot/vmlinuz-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cd /boot/
root@localhost:/boot#
```

15. `cp .config /boot/config-new-6.3.2-smp` - kopiuje plik konfiguracyjny do katalogu `/boot/`

```
root@localhost:/usr/src/linux-6.3.2# cp arch/x86/boot/bzImage /boot/vmlinuz-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp System.map /boot/System.map-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp .config /boot/vmlinuz-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cd /boot/
root@localhost:/boot#
```

16. `cd /boot/` - zmienia bieżący katalog na `/boot/`

```
root@localhost:/usr/src/linux-6.3.2# cp arch/x86/boot/bzImage /boot/vmlinuz-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp System.map /boot/System.map-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cp .config /boot/vmlinuz-new-6.3.2-smp
root@localhost:/usr/src/linux-6.3.2# cd /boot/
root@localhost:/boot#
```

17. `rm System.map` - usuwa mapę symboli.

```
root@localhost:/boot# rm System.map
root@localhost:/boot# ln -s System.map-new-6.3.2-smp System.map
root@localhost:/boot#
```

18. `ln -s System.map-new-6.3.2-smp System.map` - tworzy dowiązanie symboliczne do nowej mapy symboli

```
root@localhost:/boot# rm System.map
root@localhost:/boot# ln -s System.map-new-6.3.2-smp System.map
root@localhost:/boot#
```

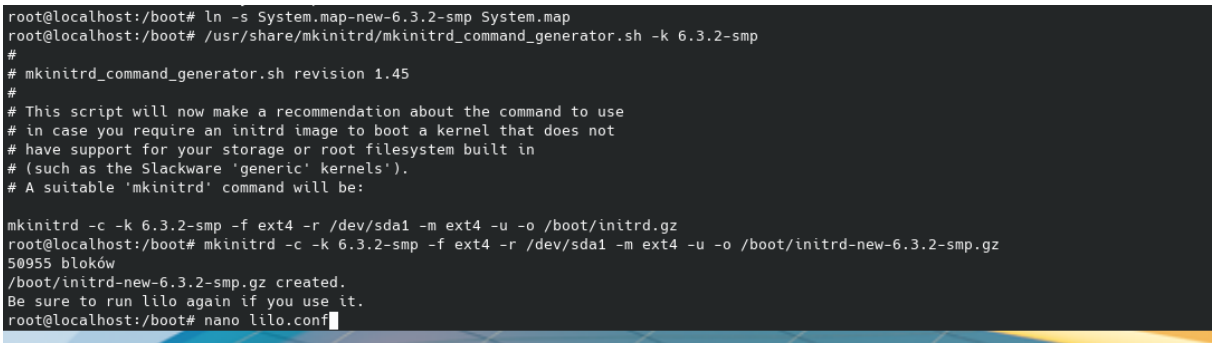
19. `/usr/share/mkinitrd/mkinitrd_command_generator.sh -k 6.3.2-smp` - generuje polecenie do utworzenia initrd dla wersji jądra `6.3.2-smp`

```
root@localhost:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 6.3.2-smp
#
# mkinitrd_command_generator.sh revision 1.45
#
# This script will now make a recommendation about the command to use
# in case you require an initrd image to boot a kernel that does not
# have support for your storage or root filesystem built in
# (such as the Slackware 'generic' kernels').
# A suitable 'mkinitrd' command will be:

mkinitrd -c -k 6.3.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
root@localhost:/boot#
```

20. `mkinitrd -c -k 6.3.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-new-6.3.2-smp.gz` - tworzy initrd dla wersji jądra `6.3.2-smp`

```
root@localhost:/boot# ln -s System.map-new-6.3.2-smp System.map
root@localhost:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 6.3.2-smp
#
# mkinitrd_command_generator.sh revision 1.45
#
# This script will now make a recommendation about the command to use
# in case you require an initrd image to boot a kernel that does not
# have support for your storage or root filesystem built in
# (such as the Slackware 'generic' kernels').
# A suitable 'mkinitrd' command will be:

mkinitrd -c -k 6.3.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
root@localhost:/boot# mkinitrd -c -k 6.3.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-new-6.3.2-smp.gz
50955 bloków
/boot/initrd-new-6.3.2-smp.gz created.
Be sure to run lilo again if you use it.
root@localhost:/boot# nano lilo.conf
```

21. `nano /etc/lilo.conf` - otwiera plik konfiguracyjny bootloadera LILO w edytorze tekstu nano



22. Dodanie rekordu w lilo.conf:

    image = /boot/vmlinuz-new-6.3.2-smp

    root = /dev/sda1

    initrd = /boot/initrd-new-6.3.2-smp.gz

    label = "nowa_metoda"

    read-only

23. Podczas wykonania lilo nastąpił Fatal error, czyli można śmiało stwierdzić, że nowa metoda zakończyła się fiaskiem i niepowodzeniem.

Wnioski

Przedstawiłem polecenia dotyczące procesu aktualizacji jądra systemu Linux. W pierwszej części pokazałem kroki związane z pobraniem i rozpakowaniem archiwum z nową wersją jądra oraz przygotowaniem pliku konfiguracyjnego. Następnie opisałem proces budowania nowego jądra oraz instalacji modułów. W kolejnych krokach przedstawiłem proces aktualizacji bootloadera LILO oraz dodanie wpisu umożliwiającego uruchomienie systemu z nowym jądrem.
W drugiej części pokazałem alternatywną metodę aktualizacji jądra, która polega na usunięciu niepotrzebnych opcji z pliku konfiguracyjnego za pomocą skryptu `streamline_config.pl`. Następnie przedstawiłem kroki analogiczne do tych z pierwszej części, ale z użyciem zmodyfikowanego pliku konfiguracyjnego.

Zgodnie z moją informacją, nowa metoda zajmuje około 50% mniej miejsca w porównaniu do starej metody. Oznacza to, że obraz jądra oraz moduły zajmują mniej miejsca na dysku. Jest to korzystne, ponieważ pozwala mi zaoszczędzić miejsce na dysku oraz przyspieszyć proces uruchamiania systemu.

Niestety, w moim przypadku nowa metoda miała niepowodzenie. Może to być spowodowane różnymi czynnikami, takimi jak niekompatybilność sprzętu lub oprogramowania z nową wersją jądra lub błędami w pliku konfiguracyjnym.

Podsumowując, przedstawiłem polecenia dotyczące procesu aktualizacji jądra systemu Linux. Pokazałem dwie metody aktualizacji: starą i nową. Nowa metoda pozwala zaoszczędzić miejsce na dysku, ale w moim przypadku miała niepowodzenie.