

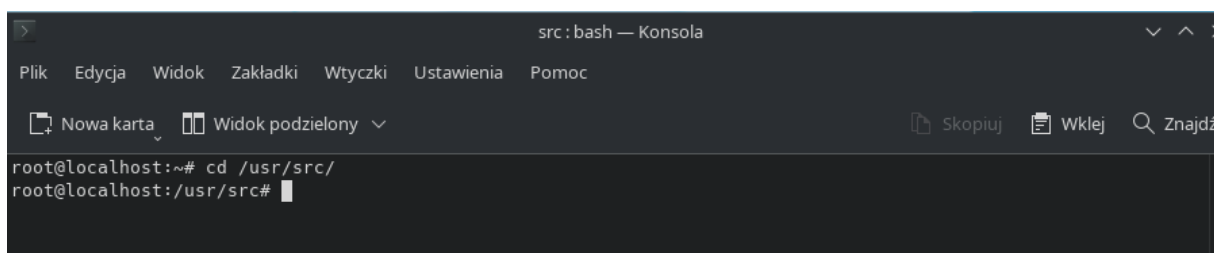
Stara Metoda

Stara metoda kompilacji jądra w systemie Slackware polegała na ręcznym konfigurowaniu pliku konfiguracyjnego jądra i ręcznym kompilowaniu jądra. W Slackware 1.0.0 z 1993 roku, plik konfiguracyjny jądra znajdował się w katalogu /usr/src/linux. Po skopiowaniu pliku konfiguracyjnego do katalogu /usr/src/linux i zmianie jego nazwy na .config, można było ręcznie zmienić opcje konfiguracji jądra i skompilować jądro.

Kroki kompilacji:

1. `cd /usr/src`
2. `wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-6.3.2.tar.xz`
3. `tar -xvpf linux-6.3.2.tar.xz`
4. `cd linux-6.3.2`
5. `zcat /proc/config.gz > .config`
6. `make localmodconfig`
7. `make -j6 bzImage`
8. `make -j6 modules`
9. `make -j6 modules_install`
10. `cp arch/x86/boot/bzImage /boot/vmlinuz-old-6.3.2-smp`
11. `cp System.map /boot/System.map-old-6.3.2-smp`
12. `cp .config /boot/config-old-6.3.2-smp`
13. `cd /boot/`
14. `rm System.map`
15. `ln -s System.map-old-6.3.2-smp System.map`
16. `/usr/share/mkinitrd/mkinitrd_command_generator.sh -k 6.3.2-smp`
17. `mkinitrd -c -k 6.3.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz`
18. `nano /etc/lilo.conf`
19. `image = /boot/vmlinuz-old-6.3.2-smp`
`root = /dev/sda1`
`initrd = /boot/initrd.gz`
`label = "old-method"`
`read-only`
20. `lilo`
21. `reboot`

Ad1.



```
src : bash — Konsola
Plik  Edycja  Widok  Zakładki  Wtyczki  Ustawienia  Pomoc
Nowa karta  Widok podzielony
root@localhost:~# cd /usr/src/
root@localhost:/usr/src#
```

Zmiana katalogu na /usr/src. Do tego katalogu w kolejnym kroku będziemy pobierać źródła kernela.

Ad 2.

```
root@localhost:usr/src# wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.3.2.tar.xz
--2023-06-06 17:59:49-- https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.3.2.tar.xz
Translacja cdn.kernel.org (cdn.kernel.org)... 146.75.121.176, 2a04:4e42:8e::432
Łączenie się z cdn.kernel.org (cdn.kernel.org)[146.75.121.176]:443... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: 136908324 (131M) [application/x-xz]
Zapis do: `linux-6.3.2.tar.xz.1'

linux-6.3.2.tar.xz.1      100%[=====] 130,57M   757KB/s   w 3m 21s

2023-06-06 18:03:10 (665 KB/s) - zapisano `linux-6.3.2.tar.xz.1' [136908324/136908324]

root@localhost:usr/src#
```

Polecenie "wget" służy do pobierania plików z sieci. W tym przypadku, pobieramy plik źródłowy jądra Linux o nazwie "linux-6.3.1.tar.xz" z podanego adresu URL.

Ad3.

```
linux-6.3.2/virt/kvm/Makefile.kvm
linux-6.3.2/virt/kvm/async_pf.c
linux-6.3.2/virt/kvm/async_pf.h
linux-6.3.2/virt/kvm/binary_stats.c
linux-6.3.2/virt/kvm/coalesced_mmio.c
linux-6.3.2/virt/kvm/coalesced_mmio.h
linux-6.3.2/virt/kvm/dirty_ring.c
linux-6.3.2/virt/kvm/eventfd.c
linux-6.3.2/virt/kvm/irqchip.c
linux-6.3.2/virt/kvm/kvm_main.c
linux-6.3.2/virt/kvm/kvm_mm.h
linux-6.3.2/virt/kvm/pfncache.c
linux-6.3.2/virt/kvm/vfio.c
linux-6.3.2/virt/kvm/vfio.h
linux-6.3.2/virt/lib/
linux-6.3.2/virt/lib/Kconfig
linux-6.3.2/virt/lib/Makefile
linux-6.3.2/virt/lib/irqbypass.c
root@localhost:usr/src# tar -xvpf linux-6.3.2.tar.xz
```

Polecenie "tar" służy do archiwizacji i rozpakowywania plików. W tym przypadku, rozpakowujemy plik "linux-6.3.2.tar.xz" używając opcji "-xvpf". W przypadku polecenia tar -xvpf linux-6.3.2.tar.xz, argumenty -xvpf mówią o tym, że chcemy rozpakować plik archiwum linux-6.3.2.tar.xz, zachowując oryginalne uprawnienia plików i wyświetlając szczegółowe informacje podczas operacji. Wynik końcowy polecenia znajduje się nad nim, ponieważ nie zdążyłam zrobić zrzutu ekranu po rozpoczęciu rozpakowywania.

Ad4.

```
root@localhost:usr/src# cd linux-6.3.2
root@localhost:usr/src/linux-6.3.2#
```

Przechodzimy do katalogu "linux-6.3.2", który został utworzony po rozpakowaniu pliku źródłowego jądra. W nim znajduje się kernel.

Ad5.

```
root@localhost:usr/src/linux-6.3.2# zcat /proc/config.gz > .config
```

Polecenie "zcat" służy do wyświetlania zawartości skompresowanego pliku. W tym przypadku, wyświetlamy zawartość pliku "/proc/config.gz" i przekierowujemy ją do pliku ".config". Wykonujemy to polecenie w celu skopiowania plików starego jądra.

Ad6.

```
root@localhost:/usr/src/linux-6.3.2# make localmodconfig
using config: '.config'
*
* Restart config...
*
*
* PCI GPIO expanders
*
AMD 8111 GPIO driver (GPIO_AMD8111) [N/m/y/?] n
BT8XX GPIO abuser (GPIO_BT8XX) [N/m/y/?] (NEW)
OKI SEMICONDUCTOR ML7213 IOH GPIO support (GPIO_ML_IOH) [N/m/y/?] n
Intel EG20T PCH/LAPIS Semiconductor IOH (ML7223/ML7831) GPIO (GPIO_PCH) [N/m/y/?] n
ACCES PCI-IDIO-16 GPIO support (GPIO_PCI_IDIO_16) [N/m/y/?] n
ACCES PCIE-IDIO-24 GPIO support (GPIO_PCIE_IDIO_24) [N/m/y/?] n
RDC R-321x GPIO support (GPIO_RDC321X) [N/m/y/?] n
*
* Hardware Monitoring support
*
Hardware Monitoring support (HWMON) [Y/n/m/?] y
  Hardware Monitoring Chip debugging messages (HWMON_DEBUG_CHIP) [N/y/?] n
*
* Native drivers
*
Abit uGuru (rev 1 & 2) (SENSORS_ABITUGURU) [N/m/y/?] n
Abit uGuru (rev 3) (SENSORS_ABITUGURU3) [N/m/y/?] n
Analog Devices AD7314 and compatibles (SENSORS_AD7314) [N/m/y/?] n
Analog Devices AD7414 (SENSORS_AD7414) [N/m/?] n
Analog Devices AD7416, AD7417 and AD7418 (SENSORS_AD7418) [N/m/?] n
Analog Devices ADM1021 and compatibles (SENSORS_ADM1021) [N/m/?] (NEW)
Analog Devices ADM1025 and compatibles (SENSORS_ADM1025) [N/m/?] n
Analog Devices ADM1026 and compatibles (SENSORS_ADM1026) [N/m/?] n
Analog Devices ADM1029 (SENSORS_ADM1029) [N/m/?] n
Analog Devices ADM1031 and compatibles (SENSORS_ADM1031) [N/m/?] n
Analog Devices ADM1177 and compatibles (SENSORS_ADM1177) [N/m/?] n
Analog Devices ADM9240 and compatibles (SENSORS_ADM9240) [N/m/?] n
Analog Devices ADT7310/ADT7320 (SENSORS_ADT7310) [N/m/y/?] n
Analog Devices ADT7410/ADT7420 (SENSORS_ADT7410) [N/m/?] n
Analog Devices ADT7411 (SENSORS_ADT7411) [N/m/?] n
Analog Devices ADT7462 (SENSORS_ADT7462) [N/m/?] n
Analog Devices ADT7470 (SENSORS_ADT7470) [N/m/?] n
Analog Devices ADT7473, ADT7475, ADT7476 and ADT7490 (SEN
```

Wyświetlanie programowe w użyciu
Wyświetlanie może być pogorszone

Polecenie "make" służy do wywoływania poleceń kompilacji z pliku Makefile. "localmodconfig" jest celem w pliku Makefile, który generuje plik konfiguracyjny dla modułów jądra na podstawie aktualnie używanych modułów w systemie. Poniżej zrzut ekranu, na którym możemy zobaczyć jak skończyło się wywołanie polecenia make localmodconfig.

```
Topstar Laptop Extras (TOPSTAR_LAPTOP) [N/m/y/?] n
Serial bus multi instantiate pseudo device driver (SERIAL_MULTI_INSTANTIATE) [N/m/?] n
Mellanox Technologies platform support (MLX_PLATFORM) [N/m/?] n
X86 Android tablet support (X86_ANDROID_TABLETS) [N/m/?] n
Intel Intelligent Power Sharing (INTEL_IPS) [N/m/y/?] n
Intel SCU PCI driver (INTEL_SCU_PCI) [Y/n/?] y
Intel SCU platform driver (INTEL_SCU_PLATFORM) [N/m/y/?] n
Intel SCU IPC utility driver (INTEL_SCU_IPC_UTIL) [N/m/y/?] n
Siemens Simatic IPC Class driver (SIEMENS_SIMATIC_IPC) [N/m/y/?] n
Winmate FM07/FM07P front-panel keys driver (WINMATE_FM07_KEYS) [N/m/y/?] n
*
* configuration written to .config
*
root@localhost:/usr/src/linux-6.3.2#
```

Ad7.

```
root@localhost:/usr/src/linux-6.3.2# make -j6 bzImage
SYNC    include/config/auto.conf.cmd
CC      arch/x86/kernel/asm-offsets.s
UPD     include/generated/asm-offsets.h
CALL    scripts/checksyscalls.sh
CC      certs/system_keyring.o
AS      arch/x86/entry/entry_32.o
CC      init/main.o
```

Kompiluje jądro Linux na podstawie konfiguracji zawartej w pliku ".config". Opcja "-j6" mówi, że kompilacja powinna być przeprowadzana równolegle na 6 wątkach. Wynikiem jest plik "bzImage", który jest jądrem Linux. Poniżej możemy zauważyć rezultat polecenia.

```
VOFFSET arch/x86/boot/compressed/../voffset.h
CC      arch/x86/boot/compressed/cmdline.o
OBJCOPY arch/x86/boot/compressed/vmlinux.bin
RELOCS  arch/x86/boot/compressed/vmlinux.relocs
CC      arch/x86/boot/compressed/error.o
CC      arch/x86/boot/compressed/early_serial_console.o
CC      arch/x86/boot/compressed/kaslr.o
CC      arch/x86/boot/compressed/acpi.o
CC      arch/x86/boot/compressed/efi.o
CC      arch/x86/boot/compressed/misc.o
LZMA    arch/x86/boot/compressed/vmlinux.bin.lzma
MKPIGGY arch/x86/boot/compressed/piggy.S
AS      arch/x86/boot/compressed/piggy.o
LD      arch/x86/boot/compressed/vmlinux
ZOFFSET arch/x86/boot/zoffset.h
OBJCOPY arch/x86/boot/vmlinux.bin
AS      arch/x86/boot/header.o
LD      arch/x86/boot/setup.elf
OBJCOPY arch/x86/boot/setup.bin
BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready (#2)
root@localhost:/usr/src/linux-6.3.2#
```

Ad 8.

```
root@localhost:/usr/src/linux-6.3.2# make -j6 modules
CALL    scripts/checksyscalls.sh
CC [M]  arch/x86/events/rapl.o
```

Kompiluje (buduje) moduły jądra Linux na podstawie konfiguracji zawartej w pliku ".config". Opcja "-j6" mówi, że kompilacja powinna być przeprowadzana równolegle na 6 wątkach. Poniżej rezultat polecenia. Budujemy jądra modułów.

```
LD [M]  net/802/stp.ko
LD [M]  net/802/mrp.ko
LD [M]  net/802/garp.ko
LD [M]  net/ipv6/ipv6.ko
LD [M]  net/8021q/8021q.ko
LD [M]  net/llc/llc.ko
LD [M]  net/wireless/cfg80211.ko
LD [M]  net/rfkill/rfkill.ko
root@localhost:/usr/src/linux-6.3.2#
```

Ad 9.

```
root@localhost:/usr/src/linux-6.3.2# make -j6 modules_install
INSTALL /lib/modules/6.3.2-smp/kernel/arch/x86/events/rapl.ko
INSTALL /lib/modules/6.3.2-smp/kernel/arch/x86/crypto/crc32-pclmul.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/video/fbdev/core/sysfillrect.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/video/fbdev/core/syscopyarea.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/video/fbdev/core/fb_sys_fops.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/video/fbdev/core/sysimgblt.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/acpi/ac.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/acpi/button.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/acpi/video.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/acpi/battery.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/char/agp/agpgart.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/char/agp/intel-agp.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/char/agp/intel-gtt.ko
```

Instaluje skompilowane moduły jądra Linux do systemu. Poniżej rezultat.

```
INSTALL /lib/modules/6.3.2-smp/kernel/net/802/garp.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/802/psnap.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/802/mrp.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/ipv6/ipv6.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/8021q/8021q.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/wireless/cfg80211.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/rfkill/rfkill.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/llc/llc.ko
DEPMOD /lib/modules/6.3.2-smp
root@localhost:/usr/src/linux-6.3.2#
```

Zawarto
root@loc

Ad 10.

```
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#
```

Kopiuje plik "bzImage" (jądro Linux) do katalogu "/boot" i nadaje mu nazwę "vmlinuz-old-6.3.2-smp".

Czyli kopiujemy obraz jądra do katalogu boot.

Ad 11.

```
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#
```

Kopiuje plik "System.map" do katalogu "/boot" i nadaje mu nazwę "System.map-old-6.3.2-smp".

Kopiuje nazwy symboli nazwanych przez kernel.

Ad 12.

```
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
root@localhost:/usr/src/linux-6.3.2# cd /boot/
```

Kopiuje plik ".config" do katalogu "/boot" i nadaje mu nazwę "config-old-6.3.2-smp". Dokładnie kopiujemy plik konfiguracyjny kernela.

Ad 13.

```
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# rm System.map
```

Przechodzimy do katalogu "/boot", aby utworzyć link symboliczny w systemie dla tablicy symboli kernela.

Ad 14.

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

Usuwa plik "System.map" , czyli starą tablicę symboli z katalogu "/boot".

Ad 15.

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

Tworzy dowiązanie symboliczne "System.map" wskazujące na plik "System.map-old-6.3.2-smp".

Ad 16.

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

Wywołuje skrypt "mkinitrd_command_generator.sh" z odpowiednimi argumentami, aby wygenerować komendę dla narzędzia mkinitrd. Skrypt generuje komendę na podstawie wersji jądra "6.3.2-smp".

Ad 17.

```
root@localhost:/boot# mkinitrd -c -k 6.3.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
50963 bloki
/boot/initrd.gz created.
Be sure to run lilo again if you use it.
root@localhost:/boot#
```

Tworzy initrd (inicjalny system plików RAM) dla wskazanej wersji jądra. Parametry określają m.in. system plików (ext4), urządzenie korzenia ("/dev/sda1") i plik wyjściowy ("/boot/initrd.gz").

Ad 18 i 19.

```
root@localhost:/boot# nano /etc/lilo.conf
root@localhost:/boot# lilo
Warning: LBA32 addressing assumed
Added Slackware_15 *
Added kernel-custom +
Added old-method +
One warning was issued.
root@localhost:/boot#
```

```
#vga=771
# VESA framebuffer console @ 640x480x64k
#vga=785
# VESA framebuffer console @ 640x480x32k
#vga=784
# VESA framebuffer console @ 640x480x256
#vga=769
# End LILO global section
# Linux bootable partition config begins
image = /boot/vmlinuz
root = /dev/sda1
label = "Slackware 15"
read-only

image = /boot/vmlinuz-custom-6.3.1-smp
root = /dev/sda1
initrd = /boot/initrd-custom-6.3.1-smp.gz
label = "kernel-custom"
read-only

image = /boot/vmlinuz-old-6.3.2-smp
root = /dev/sda1
initrd = /boot/initrd.gz
label = "old-method"
read-only
# Linux bootable partition config ends
```

Otwiera plik konfiguracyjny LILO (Linux Loader) w edytorze tekstowym Nano.

W edytorze Nano należy wprowadzić następujące linie w pliku `/etc/lilo.conf`, aby dodać nowy wpis do bootloadera lilo:

```
image = /boot/vmlinuz-old-6.3.2-smp
```

```
root = /dev/sda1
```

```
initrd = /boot/initrd.gz
```

```
label = "old-method"
```

```
read-only
```

Linie te definiują nowy wpis w konfiguracji LILO, wskazujący na starą wersję jądra, partycję root, plik initrd i etykietę dla tego wpisu.

Ad 20.

```
root@localhost:/boot# nano /etc/lilo.conf
root@localhost:/boot# lilo
Warning: LBA32 addressing assumed
Added Slackware_15 *
Added kernel-custom +
Added old-method +
One warning was issued.
root@localhost:/boot#
```

Aktualizuje konfigurację LILO na podstawie zmian dokonanych w pliku `/etc/lilo.conf`.

Ad 21.

Reboot: Przeładowuje system, co powoduje restart komputera.

Nowa metoda

Musimy przygotować katalog src do ponownego skompilowania jądra nową metodą.

Polecenia:

1. `cd usr/src/`
2. `cd linux-6.3.2`
3. `rm -R Linux-6.3.2`
4. `tar -xvpf linux-6.3.2.tar.xz`
5. `cd linux-6.3.2`

Następnie zaczynamy na nowo kompilację jądra:

6. `cp /boot/config .config`
7. `./scripts/kconfig/streamline_config.pl > config_strip`
8. `mv .config config.bak`
9. `mv config_strip .config`
10. `make oldconfig`
11. `make -j6 bzImage`
12. `make -j6 modules`
13. `make -j6 modules_install`
14. `cp arch/x86/boot/bzImage /boot/vmlinuz-new-6.3.2-smp`
15. `cp System.map /boot/System.map-new-6.3.2-smp`
16. `cp .config /boot/config-new-6.3.2-smp`
17. `cd /boot/`
18. `rm System.map`
19. `ln -s System.map-new-6.3.2-smp System.map`
20. `/usr/share/mkinitrd/mkinitrd_command_generator.sh -k 6.3.2-smp`
21. `mkinitrd -c -k 6.3.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-new-6.3.2-smp.gz`
22. `nano /etc/lilo.conf`
image = /boot/vmlinuz-new-6.3.2-smp
root = /dev/sda1
initrd = /boot/initrd.gz
label = "new-method"
read-only
23. `lilo`
24. `reboot`

Ad 1.

```
root@localhost:/usr/src/linux-6.3.2# cd /usr/src/
```

Przechodzimy do katalogu usr/src.

Ad 2 i Ad 3.

```
root@localhost:/usr/src# cd linux-6.3.2
root@localhost:/usr/src/linux-6.3.2# rm -R linux-6.3.2
```

Usuwamy folder w którym znajduje się poprzednia konfiguracja jądra stara metodą.

Ad 4 i 5.

```
root@localhost:/usr/src/linux-6.3.2# tar -xvpf linux-6.3.2.tar.xz
```

```
linux-6.3.2/virt/kvm/dirty_ring.c
linux-6.3.2/virt/kvm/eventfd.c
linux-6.3.2/virt/kvm/irqchip.c
linux-6.3.2/virt/kvm/kvm_main.c
linux-6.3.2/virt/kvm/kvm_mm.h
linux-6.3.2/virt/kvm/pfnocache.c
linux-6.3.2/virt/kvm/vfio.c
linux-6.3.2/virt/kvm/vfio.h
linux-6.3.2/virt/lib/
linux-6.3.2/virt/lib/Kconfig
linux-6.3.2/virt/lib/Makefile
linux-6.3.2/virt/lib/irqbypass.c
root@localhost:/usr/src# cd linux-6.3.2
```

Rozpakowujemy folder pobrany w starej metodzie i po rozpakowaniu przechodzimy do folderu z plikami linux-6.3.2.

Ad 6.

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

Kopiujemy plik config znajdujący się w katalogu /boot do bieżącego katalogu, a następnie zmieniamy jego nazwę na .config.

Ad 7.

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

Uruchamiamy skrypt streamline_config.pl, który generuje zoptymalizowaną wersję pliku konfiguracyjnego i zapisuje go do pliku config_strip.

Ad 8.

```
using 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
```

Przenosi plik .config do config.bak, nadpisując go, jeśli już istnieje.

Ad 9.

```
root@localhost:/usr/src/linux-6.3.2# mv config_strip .config
root@localhost:/usr/src/linux-6.3.2#
```

Przenosi plik config_strip do .config, nadpisując go, jeśli już istnieje.

Ad 10.

```
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
```

```

Test floating point operations in kernel space (TEST_FPU) [N/m/y/?] n
Test clocksource watchdog in kernel space (TEST_CLOCKSOURCE_WATCHDOG) [N/m/y/?] n
#
# configuration written to .config
#

```

Uruchamia proces konfiguracji, który sprawdza, czy istnieją nowe opcje konfiguracyjne w pliku .config, i pyta użytkownika o ewentualne zmiany.

Ad 11.

```

root@localhost:/usr/src/linux-6.3.2# make -j6 bzImage
SYSHDR arch/x86/include/generated/uapi/asm/unistd_32.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_x32.h
SYSTBL arch/x86/include/generated/asm/syscalls_32.h
WRAP arch/x86/include/generated/uapi/asm/bpf_perf_event.h
WRAP arch/x86/include/generated/uapi/asm/errno.h
WRAP arch/x86/include/generated/uapi/asm/fcntl.h
WRAP arch/x86/include/generated/uapi/asm/ioctl.h
WRAP arch/x86/include/generated/uapi/asm/ioctls.h

```

Rozpoczyna proces kompilacji jądra systemu Linux (pliku bzImage) z wykorzystaniem 6 wątków. Poniżej rezultat wykonania polecenia.

```

CC arch/x86/boot/compressed/misc.o
LZMA arch/x86/boot/compressed/vmlinux.bin.lzma
MKPIGGY arch/x86/boot/compressed/piggy.S
AS arch/x86/boot/compressed/piggy.o
LD arch/x86/boot/compressed/vmlinux
ZOFFSET arch/x86/boot/zoffset.h
OBJCOPY arch/x86/boot/vmlinux.bin
AS arch/x86/boot/header.o
LD arch/x86/boot/setup.elf
OBJCOPY arch/x86/boot/setup.bin
BUILD arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready (#1)
root@localhost:/usr/src/linux-6.3.2#

```

Ad 12.

```

root@localhost:/usr/src/linux-6.3.2# make -j6 modules
CALL scripts/checksyscalls.sh
LDS scripts/module.lds
CC [M] arch/x86/events/rapl.o
CC [M] sound/core/sound.o
CC [M] sound/core/init.o
CC [M] net/802/p8022.o
AS [M] arch/x86/crypto/crc32-pclmul_asm.o
CC [M] sound/core/memory.o
CC [M] arch/x86/crypto/crc32-pclmul_glue.o
CC [M] drivers/video/fbdev/core/sysfillrect.o

```

Rozpoczyna proces kompilacji modułów jądra systemu Linux z wykorzystaniem 6 wątków. Poniżej rezultaty polecenia.

```

LD [M] net/802/mrp.ko
LD [M] net/802/garp.ko
LD [M] net/ipv6/ipv6.ko
LD [M] net/8021q/8021q.ko
LD [M] net/wireless/cfg80211.ko
LD [M] net/llc/llc.ko
LD [M] net/rfkill/rfkill.ko
root@localhost:/usr/src/linux-6.3.2#

```

Ad 13.

```
LD [M] net/rfkill/rfkill.ko
root@localhost:/usr/src/linux-6.3.2# make -j6 modules_install
INSTALL /lib/modules/6.3.2-smp/kernel/arch/x86/events/rapl.ko
INSTALL /lib/modules/6.3.2-smp/kernel/arch/x86/crypto/crc32-pclmul.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/video/fbdev/core/sysfillrect.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/video/fbdev/core/fb_sys_fops.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/video/fbdev/core/syscopyarea.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/video/fbdev/core/sysimgblt.ko
INSTALL /lib/modules/6.3.2-smp/kernel/drivers/acpi/ac.ko
```

Instaluje skompilowane moduły jądra systemu Linux. Rezultaty poniżej.

```
INSTALL /lib/modules/6.3.2-smp/kernel/net/802/mrp.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/ipv6/ipv6.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/8021q/8021q.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/wireless/cfg80211.ko
INSTALL /lib/modules/6.3.2-smp/kernel/net/llc/llc.ko
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#
```

Ad 14.

```
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
```

Kopiuje skompilowane jądro (bzImage) do katalogu /boot i nadaje mu nazwę vmlinuz-new-6.3.2-smp.

Ad 15.

```
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/config-new-6.3.2-smp
```

Kopiuje plik System.map do katalogu /boot i nadaje mu nazwę System.map-new-6.3.2-smp.

Ad 16.

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

Kopiuje plik .config do katalogu /boot i nadaje mu nazwę config-new-6.3.2-smp.

Ad 17.

```
root@localhost:/usr/src/linux-6.3.2# cd /boot/
```

Zmienia bieżący katalog na /boot.

Ad 18.

```
root@localhost:/boot# rm System.map
```

Usuwa plik System.map z bieżącego katalogu boot.

Ad 19.

```
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
```

Tworzy dowiązanie symboliczne o nazwie System.map, które wskazuje na plik System.map-new-6.3.2-smp.

Ad 20.

```
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
```

Uruchamia skrypt `mkinitrd_command_generator.sh` z odpowiednimi parametrami, aby wygenerować polecenie `mkinitrd`.

Ad 21.

```
root@localhost:/boot# mkinitrd -c -k 6.3.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
50963 bloki
/boot/initrd.gz created.
Be sure to run lilo again if you use it.
```

Tworzy nowy obraz `initrd` o nazwie `initrd.gz`, który jest potrzebny podczas uruchamiania systemu.

Ad 22.

```
# End Lilo global section
# Linux bootable partition config begins
image = /boot/vmlinuz
root = /dev/sda1
label = "Slackware 15"
read-only

image = /boot/vmlinuz-custom-6.3.1-smp
root = /dev/sda1
initrd = /boot/initrd-custom-6.3.1-smp.gz
label = "kernel-custom"
read-only

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

image = /boot/vmlinuz-new-6.3.2-smp
root = /dev/sda1
initrd = /boot/initrd.gz
label = "new-method"
read-only

# Linux bootable partition config ends
```

Otwiera plik `lilo.conf` w edytorze tekstowym `nano`, umożliwiając wprowadzanie zmian w konfiguracji bootloadera. W pliku `lilo.conf` definiujemy ścieżkę do pliku jądra (`vmlinuz-new-6.3.2-smp`), ścieżkę do partycji głównej (`/dev/sda1`), do obrazu `initrd` (`initrd.gz`). Definiuje etykietę, która identyfikuje nową konfigurację, oraz że partycja jest montowana w trybie tylko do odczytu.

Ad 23.

```
root@localhost:/boot# nano /etc/lilo.conf
root@localhost:/boot# lilo
Warning: LBA32 addressing assumed
Added Slackware_15 *
Added kernel-custom +
Added old-method +
Added new-method +
One warning was issued.
root@localhost:/boot#
```

Uruchamia program lilo, który zapisuje konfigurację bootloadera i instaluje go w sektorze rozruchowym dysku.

Wnioski:

Uważam, że proces kompilacji zarówno jedną jak i drugą metodą jest zbyt czasochłonny. Moim zdaniem kompilacja jądra nową metodą jest dla mnie lepsza, ponieważ udało mi się poznać ten sposób podczas zajęć i jest dla mnie bardziej zrozumiały. Po kompilacji zarówno starą jak i nową metodą po dłuższej chwili oczekiwania system włączył się ponownie. Nie byłam jednak w stanie stwierdzić czy uruchomił się na nowej wersji jądra, ponieważ przez niestabilną wersję virtualboxa system bardzo się zacinął.

Link do folderu źródłowego z plikami starej metody: https://umcso365-my.sharepoint.com/:u:/g/personal/303883_office_umcs_pl/EQCfM2MHuz9Egw8YqNruvGsBrg85cBSKRQkzjoC8b2enHQ?e=oUMcnx

Link do folderu źródłowego z plikami nowej metody: https://umcso365-my.sharepoint.com/:u:/g/personal/303883_office_umcs_pl/Ef4zBaU3csdEnweRSnEGknUBNu8wOJrmphclzpoY5-ShYw?e=i1CwHm