

Kompilacja jądra

Szymon Jędrych

1.06.2022

Linux

Przygotowanie plików jądra

Aby skompilować jądro, pierwszym krokiem jest pobranie aktualnej wersji. Na dzień 1 czerwca 2022 roku najnowszą wersją jest **5.18**. Jądro pobieramy ze strony <https://kernel.org/>. Wymienioną wersję pobieramy w archiwum o rozszerzeniu `.tar` ze strony <https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.tar.xz>. Jeśli posiadamy wcześniej przygotowane jądro możemy wykorzystać je do kompilacji bez pobierania nowego.

Pliki jądra będziemy przechowywać w katalogu `/usr/src`, dlatego przechodzimy tam komendą:

```
cd /usr/src
```

Po przejściu do podanego katalogu używamy polecenia `wget` do pobrania jądra linuxa:

```
wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.tar.xz
```



```
root@slack:~# cd /usr/src
root@slack:~# cd /usr/src# wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.tar.xz
--2022-05-29 17:38:05-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.tar.xz
Transakcja z cdn.kernel.org (cdn.kernel.org)... 151.101.13.176: 2004:4e42:3::432
Odebranie z cdn.kernel.org (cdn.kernel.org)[151.101.13.176:443... połączono.
Odebranie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: 129790264 (124M) [application/x-xz]
Zapis do: 'linux-5.18.tar.xz'
linux-5.18.tar.xz 100%[=====] 123,78M 11,1MB/s w 8,5s
2022-05-29 17:38:14 (14,6 MB/s) - zapisano 'linux-5.18.tar.xz' [129790264/129790264]
root@slack:~# cd /usr/src# S
```

Rysunek 1: Pobieranie plików źródłowych

Następnie należy pliki wypakować poleceniem `tar` z argumentami `-xvf`:

```
tar -xvf linux-5.18.tar.xz
```

```

linux-5.18/tools/uml/Makefile
linux-5.18/tools/uml/dell-smbios-example.c
linux-5.18/usr/
linux-5.18/usr/.gitignore
linux-5.18/usr/Kconfig
linux-5.18/usr/Makefile
linux-5.18/usr/default_cpio_list
linux-5.18/usr/gen_init_cpio.c
linux-5.18/usr/gen_initramfs.sh
linux-5.18/usr/include/
linux-5.18/usr/include/.gitignore
linux-5.18/usr/include/Makefile
linux-5.18/usr/include/headers_check.pl
linux-5.18/usr/initramfs_data.S
linux-5.18/virt/
linux-5.18/virt/Makefile
linux-5.18/virt/kvm/
linux-5.18/virt/kvm/Kconfig
linux-5.18/virt/kvm/Makefile.kvm
linux-5.18/virt/kvm/async_pf.c
linux-5.18/virt/kvm/async_pf.h
linux-5.18/virt/kvm/binary_stats.c
linux-5.18/virt/kvm/coalesced_mmio.c
linux-5.18/virt/kvm/coalesced_mmio.h
linux-5.18/virt/kvm/dirty_ring.c
linux-5.18/virt/kvm/eventfd.c
linux-5.18/virt/kvm/irqchip.c
linux-5.18/virt/kvm/kvm_main.c
linux-5.18/virt/kvm/kvm_mmu.h
linux-5.18/virt/kvm/pfncache.c
linux-5.18/virt/kvm/vfio.c
linux-5.18/virt/kvm/vfio.h
linux-5.18/virt/lib/
linux-5.18/virt/lib/Kconfig
linux-5.18/virt/lib/Makefile
linux-5.18/virt/lib/irqbypass.c
root@slack:/usr/src# _

```

Rysunek 2: Wypakowywanie plików źródłowych

Po wypakowaniu plików jesteśmy gotowi do kompilacji jądra.

1 Metoda stara - tworzenie pliku konfiguracyjnego

Należy skopiować konfigurację aktualnego kernela do pliku `.config`

```

root@slack:/usr/src/linux-5.18# zcat /proc/config.gz > .config
root@slack:/usr/src/linux-5.18# ls -la | grep .config
-rw-r--r-- 1 root root 59 maj 22 21:52 .config
-rw-r--r-- 1 root root 237798 maj 29 19:54 .config
-rw-r--r-- 1 root root 555 maj 22 21:52 Kconfig
root@slack:/usr/src/linux-5.18# _

```

Rysunek 3: Skopiowanie aktualnego configu

Po utworzeniu configu używamy komendy `make localmodconfig` do stworzenia pliku konfiguracyjnego. Zostaniemy zapytani o to konfigurację poszczególnych modułów jądra, wszystko zostawiamy domyślnie. Wynik komendy:

```

Perform selftest on siphash functions (TEST_SIPHASH) (N/n/y/?) (NEW)
Perform selftest on IDA functions (TEST_IDA) (N/n/y/?) n
Test module loading with 'hello world' module (TEST_LWM) (N/n/?) n
Test module for compilation of bitops operations (TEST_BITOPS) (N/n/?) n
Test module for stress/performance analysis of umalloc allocator (TEST_UMALLOC) (N/n/?) n
Test user/kernel boundary protections (TEST_USER_COPY) (N/n/?) n
Test BPF filter functionality (TEST_BPF) (N/n/?) n
Test blackhole netdev functionality (TEST_BLACKHOLE_DEV) (N/n/?) n
Test find_bit functions (FIND_BIT_BENCHMARK) (N/n/y/?) n
Test firmware loading via userspace interface (TEST_FIRMWARE) (N/n/y/?) n
sysctl test driver (TEST_SYSCTL) (N/n/y/?) n
udelay test driver (TEST_UDELAY) (N/n/y/?) n
Test static keys (TEST_STATIC_KEYS) (N/n/?) n
kmod stress tester (TEST_KMOD) (N/n/?) n
Test memcat_p() helper function (TEST_MEMCAT_P) (N/n/y/?) n
Test heap/pages initialization (TEST_REINIT) (N/n/y/?) n
Test freeing pages (TEST_FREE_PAGES) (N/n/y/?) n
Test floating point operations in kernel space (TEST_FPU) (N/n/y/?) n
Test clocksource watchdog in kernel space (TEST_CLOCKSOURCE_WATCHDOG) (N/n/y/?) n
configuration written to .config
root@slack:/usr/src/linux-5.18#

```

Rysunek 4: Zakończona konfiguracja jądra

Jesteśmy gotowi do kompilacji jądra. Użyta do tego została komenda `make::`

```

LD [M] drivers/usb/host/ohci-hcd.ko
LD [M] drivers/usb/host/ohci-pci.ko
LD [M] drivers/usb/host/ohci-hcd.ko
LD [M] drivers/usb/host/ohci-pci.ko
LD [M] drivers/video/fbdev/core/fb_sys_fops.ko
LD [M] drivers/video/fbdev/core/syscopyarea.ko
LD [M] drivers/video/fbdev/core/sysfillrect.ko
LD [M] drivers/video/fbdev/core/sysimgblt.ko
LD [M] drivers/virt/vboxguest/vboxguest.ko
LD [M] net/802/garp.ko
LD [M] net/802/mrp.ko
LD [M] net/802/p8022.ko
LD [M] net/802/p802p.ko
LD [M] net/802/stp.ko
LD [M] net/8021q/8021q.ko
LD [M] net/ipv6/ipv6.ko
LD [M] net/rxllz/rxllz.ko
LD [M] net/wireless/cfg80211.ko
LD [M] net/llc/llc.ko
LD [M] sound/ac97_bus.ko
LD [M] sound/core/snd-pcm.ko
LD [M] sound/core/snd.ko
LD [M] sound/pci/ac97/snd-ac97-codec.ko
LD [M] sound/pci/snd-intel8x0.ko
LD [M] sound/core/snd-timer.ko
LD [M] sound/soundcore.ko
MKPIGGY arch/x86/boot/compressed/piggy.S
AS arch/x86/boot/compressed/piggy.o
LD arch/x86/boot/compressed/umlinux
ZDPPSET arch/x86/boot/zdppset.h
OBJCOPY arch/x86/boot/umlinux.bin
AS arch/x86/boot/header.o
LD arch/x86/boot/setup.elf
OBJCOPY arch/x86/boot/setup.bin
BUILD arch/x86/boot/hzImage
Kernel: arch/x86/boot/hzImage is ready (#1)
root@slack:/usr/src/linux-5.18#

```

Rysunek 5: Zakończona kompilacja jądra

Po kompilacji jądra należy podobnie skompilować moduły komendą `make modules`:

```

root@slack:/usr/src/linux-5.18# make modules
CALL scripts/checksyscalls.sh
CALL scripts/atomic/check-atomics.sh
root@slack:/usr/src/linux-5.18# _

```

Rysunek 6: Zakończona kompilacja modułów

Wykonujemy również komendę `make modules_install` w celu wygenerowania komendy generującej ramdisk :

```

INSTALL /lib/modules/5.18.0-snp/kernel/drivers/i2c/i2c-core.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/input/eudev.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/input/joydev.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/input/mouse/psmouse.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/input/serio/serio_raw.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/net/ethernet/amd/pcnet32.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/net/mii.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/powercap/intel_rapl_common.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/powercap/intel_rapl_rsr.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/usb/host/ehci-hcd.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/usb/host/ehci-pci.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/usb/host/ohci-hcd.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/usb/host/ohci-pci.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/video/fbdev/core/fb_sys_fops.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/video/fbdev/core/syscopyarea.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/video/fbdev/core/sysfillrect.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/video/fbdev/core/sysimgblt.ko
INSTALL /lib/modules/5.18.0-snp/kernel/drivers/virt/vboxguest/vboxguest.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/802/garp.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/802/rarp.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/802/p8022.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/802/psnap.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/802/stp.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/802/q/8021q.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/ipv6/ipv6.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/llc/llc.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/rfkill/rfkill.ko
INSTALL /lib/modules/5.18.0-snp/kernel/net/wireless/cfg80211.ko
INSTALL /lib/modules/5.18.0-snp/kernel/sound/ac97_bus.ko
INSTALL /lib/modules/5.18.0-snp/kernel/sound/core/snd-pcm.ko
INSTALL /lib/modules/5.18.0-snp/kernel/sound/core/snd-timer.ko
INSTALL /lib/modules/5.18.0-snp/kernel/sound/core/snd.ko
INSTALL /lib/modules/5.18.0-snp/kernel/sound/pci/ac97/snd-ac97-codec.ko
INSTALL /lib/modules/5.18.0-snp/kernel/sound/pci/snd-intel8x0.ko
INSTALL /lib/modules/5.18.0-snp/kernel/sound/soundcore.ko
DEPMOD /lib/modules/5.18.0-snp
root@slack:/usr/src/linux-5.18#

```

Rysunek 7: Komenda make install

Następnie należy skopiować pliki potrzebne do uruchomienia nowego jądra do katalogu `/boot`:

```

root@slack:/usr/src/linux-5.18# cp arch/x86/boot/bzImage /boot/vmlinuz-metodastara-5.18-snp
root@slack:/usr/src/linux-5.18# cp System.map /boot/System.map-metodastara-5.18-snp
root@slack:/usr/src/linux-5.18# cp .config /boot/config-metodastara-5.18-snp
root@slack:/usr/src/linux-5.18#

```

Rysunek 8: Kopiowanie plików do katalogu boot

Po kopiowaniu jesteśmy gotowi do zlinkowania pliku `System.map`:

```

root@slack:/usr/src/linux-5.18# cd /boot
root@slack:/boot# ls -la | grep System.map
lrwxrwxrwx 1 root root    31 kwi 25 19:06 System.map -> System.map-huge-snp-5.15.27-snp
-rw-r--r-- 1 root root 3883385 mar  9 02:44 System.map-generic-5.15.27
-rw-r--r-- 1 root root 4020483 mar  9 04:00 System.map-generic-snp-5.15.27-snp
-rw-r--r-- 1 root root 5333639 mar  9 02:41 System.map-huge-5.15.27
-rw-r--r-- 1 root root 5473713 mar  9 03:55 System.map-huge-snp-5.15.27-snp
-rw-r--r-- 1 root root 5453466 maj 29 20:24 System.map-metodastara-5.18-snp
root@slack:/boot# rm System.map
root@slack:/boot# ln -s System.map-metodastara-5.18-snp System.map
root@slack:/boot#

```

Rysunek 9: Podmianienie pliku System.map

Po linkowaniu możemy wygenerować komendę służącą do wygenerowania ramdisku:

```

root@slack:/usr/src/linux-5.18# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.0-snp
#
# 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 5.18.0-snp -f ext4 -r /dev/sda1 -n ext4 -u -o /boot/initrd.gz
root@slack:/usr/src/linux-5.18#

```

Rysunek 10: Generowanie komendy do generacji ramdisk

Używamy teraz wygenerowanej komendy do utworzenia ramdisku:

`mkinitrd -c -k 5.18.0-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz`

```
root@slack:/usr/src/linux-5.18# mkinitrd -c -k 5.18.0-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/
initrd.gz
49030 bloków
/boot/initrd.gz created.
Be sure to run lilo again if you use it.
root@slack:/usr/src/linux-5.18# _
```

Rysunek 11: Generowanie ramdisk

Aby ukazać wygenerowany plik `lilo.conf` używamy `cat /etc/lilo.conf`:

```
# End LILO global section
# Linux bootable partition config begins
image = /boot/vmlinuz
root = /dev/sda1
label = "Slackware 15.0"
read-only
# Linux bootable partition config ends
root@slack:/usr/src/linux-5.18# _
```

Rysunek 12: Wygenerowany plik `lilo.conf`

Edytujemy plik w celu ustawienia nowego kernela i uruchamiamy komendę `lilo`:

```
# End LILO global section
# Linux bootable partition config begins
image = /boot/vmlinuz-metodastara-5.18-smp
root = /dev/sda1
initrd = /boot/initrd.gz
label = "metoda stara"
read-only
# Linux bootable partition config ends
root@slack:/usr/src/linux-5.18# is /boot
README, initrd
System.map
config-huge-smp-5.15.27-smp tuxlogo.bmp
config-metodastara-5.18-smp tuxlogo.dat
System.map-generic-5.15.27
System.map-generic-smp-5.15.27-smp
System.map-huge-5.15.27
System.map-huge-smp-5.15.27-smp
System.map-metodastara-5.18-smp
boot.8890
boot_message.txt
config
config-generic-5.15.27
config-generic-smp-5.15.27-smp
config-huge-5.15.27
config-huge-smp-5.15.27-smp
grub/
initrd-tree/
initrd.gz
inside.bmp
inside.dat
map
onlukluc.bmp
onlukluc.dat
slack.bmp
onlinux
onlinux-generic
onlinux-generic-5.15.27
onlinux-generic-smp
onlinux-generic-smp-5.15.27-smp
onlinux-huge
onlinux-huge-5.15.27
onlinux-huge-smp
onlinux-huge-smp-5.15.27-smp
onlinux-metodastara-5.18-smp
root@slack:/usr/src/linux-5.18# lilo
Warning: LBR32 addressing assumed
Warning: Unable to determine video adapter in use in the present system.
Warning: Video adapter does not support VESA BIOS extensions needed for
display of 256 colors. Boot loader will fall back to TEXT only operation.
Added metoda_stara * *
3 warnings were issued.
root@slack:/usr/src/linux-5.18# _
```

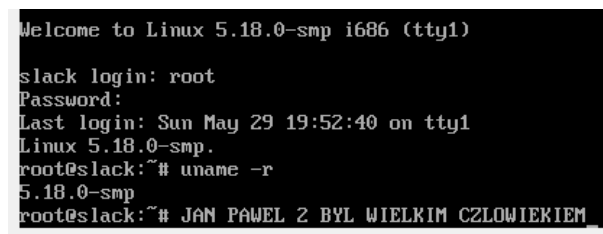
Rysunek 13: Edycja i uruchomienie `lilo`

Jesteśmy gotowi zrestartować maszynę. Wpis pojawił się przy uruchamianiu:



Rysunek 14: Uruchamianie nowego jądra

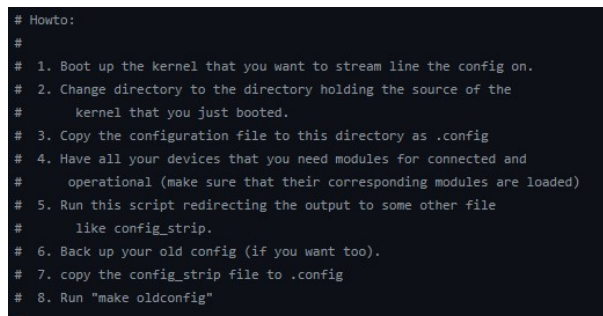
Po zalogowaniu widzimy, że nowe jądro zostało prawidłowo uruchomione:



Rysunek 15: Sprawdzenie działania jądra

2 Metoda nowa - streamline_{config}.pl

Należy wygenerować plik konfiguracyjny zgodnie z instrukcją:



Rysunek 16: Instrukcja wygenerowania skryptu

Wykonujemy kilka komend niezbędnych do przygotowania środowiska:

```
root@slack:/usr/src/linux-5.10# wget https://raw.githubusercontent.com/torvalds/linux/master/scripts/kconfig/streamline_config.pl
--2022-05-29 21:04:39-- https://raw.githubusercontent.com/torvalds/linux/master/scripts/kconfig/streamline_config.pl
Translacja raw.githubusercontent.com (raw.githubusercontent.com)... 195.159.188.133, 195.159.111.133, 195.159.109.133, ...
*zmiana n/a z raw.githubusercontent.com (raw.githubusercontent.com)195.159.188.133:443... po#wzrost.
*znajdź HTTP us#ano, oczekiwanie na odpowiedź... 200 OK
*znajdź: 16820 (168) (text/plain)
*zapis do: 'streamline_config.pl'
streamline_config.pl 1 100[=====] 15,43K --,KB/s w 0,002s
2022-05-29 21:04:39 (0,65 MB/s) - zapisano 'streamline_config.pl' (16820/16820)
root@slack:/usr/src/linux-5.10# cp streamline_config.pl ./scripts/kconfig/streamline_config.pl
root@slack:/usr/src/linux-5.10# cp ./streamline_config.pl ./scripts/kconfig/streamline_config.pl
root@slack:/usr/src/linux-5.10# ./scripts/
Display all 104 possibilities (y or n)
root@slack:/usr/src/linux-5.10# ./scripts/
Display all 104 possibilities (y or n)
root@slack:/usr/src/linux-5.10# ./scripts/kconfig/streamline_config.pl > config_strip
Using config: '.config'
root@slack:/usr/src/linux-5.10#
```

Rysunek 17: Przygotowanie środowiska

Po konfiguracji wykonujemy make oldconfig jak kaže instrukcja:

```
Test functions located in the string_helpers module at runtime (TEST_STRING_HELPERS) [N/n/y/?] n
Test strcpy() family of functions at runtime (TEST_STRCPY) [N/n/y/?] n
Test kstrto() family of functions at runtime (TEST_KSTRTO) [N/n/y/?] n
Test printf() family of functions at runtime (TEST_PRINTF) [N/n/y/?] n
Test scanf() family of functions at runtime (TEST_SCANF) [N/n/y/?] n
Test bitmap_*() family of functions at runtime (TEST_BITMAP) [N/n/y/?] n
Test functions located in the uuid module at runtime (TEST_UUID) [N/n/y/?] n
Test the Xorray code at runtime (TEST_XORRAY) [N/n/y/?] n
Perform selftest on resizable hash table (TEST_HASHABLE) [N/n/y/?] n
Perform selftest on siphash functions (TEST_SIPHASH) [N/n/y/?] (NEW)
Perform selftest on IDA functions (TEST_IDA) [N/n/y/?] n
Perform selftest on priority array manager (TEST_PRIORITY) [N/n/y/?] n
Test module loading with 'hello world' module (TEST_LKM) [N/n/y/?] n
Test module for compilation of bitops operations (TEST_BITOPS) [N/n/y/?] n
Test module for stress/performance analysis of umalloc allocator (TEST_UMALLOC) [N/n/y/?] n
Test user/kernel boundary protections (TEST_USER_COPY) [N/n/y/?] n
Test BPF filter functionality (TEST_BPF) [N/n/y/?] n
Test blackhole netdev functionality (TEST_BLACKHOLE_DEV) [N/n/y/?] n
Test find_bit functions (FIND_BIT_BENCHMARK) [N/n/y/?] n
Test firmware loading via userspace interface (TEST_FIRMWARE) [N/n/y/?] n
sysctl test driver (TEST_SYSCTL) [N/n/y/?] n
udelay test driver (TEST_UDELAY) [N/n/y/?] n
Test static keys (TEST_STATIC_KEYS) [N/n/y/?] n
kmod stress tester (TEST_KMOD) [N/n/y/?] n
Test memcat_p() helper function (TEST_MEMCAT_P) [N/n/y/?] n
Perform selftest on object aggregation manager (TEST_OBAGG) [N/n/y/?] n
Test heap/page initialization (TEST_MEMINIT) [N/n/y/?] n
Test freeing pages (TEST_FREE_PAGES) [N/n/y/?] n
Test floating point operations in kernel space (TEST_FPU) [N/n/y/?] n
Test clocksource watchdog in kernel space (TEST_CLOCKSOURCE_WATCHDOG) [N/n/y/?] n
+ configuration written to .config
root@slack:/usr/src/linux-5.10#
```

Rysunek 18: Wykonanie komendy make oldconfig

Następnie ponownie wykonujemy kompilację jądra i modułów oraz ich instalację komendą
make make modules make modules_install

```
root@slack:/usr/src/linux-5.10# make && make modules && make modules_install
```

Rysunek 19: Kompilacja i instalacja jądra i modułów

[illegible]

Rysunek 20: Wynik kompilacji i instalacji jądra/modułów

Wykonujemy następnie ten sam krok co w poprzedniej metodzie, to znaczy kopiujemy niezbędne pliki do katalogu `/boot` i tworzymy dowiązanie `System.map`:

```
root@jack:/usr/src/linux-5.18# cp arch/x86/boot/bzImage /boot/vmlinuz-metodanowa-5.18-smp
root@jack:/usr/src/linux-5.18# cp System.map /boot/System.map-metodanowa-5.18-smp
root@jack:/usr/src/linux-5.18# cp .config /boot/config-metodanowa-5.18-smp
root@jack:/usr/src/linux-5.18# cp /usr/src/linux-5.18/arch/x86/boot/compressed/vmlinuz-5.18-smp /boot/System.map-metodanowa-5.18-smp
root@jack:/usr/src/linux-5.18# ln -s /boot/System.map-metodanowa-5.18-smp /boot/System.map
ln: nie udało się utworzyć dowiązania symbolicznego 'System.map': Plik istnieje
root@jack:/usr/src/linux-5.18# ln -s /boot/System.map-metodanowa-5.18-smp /boot/System.map
```

Rysunek 21: Przekopiowanie plików konfiguracyjnych i stworzenie dowiązań

Analogicznie tworzymy też ramdisk:

```
root@rack:/usr/src/linux-4.10.0-rc2/share# kbuild/mkinitrd mkinitrd_command_generator.sh -k 5.10.0-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 Shadowware "generic" kernels').
A suitable 'mkinitrd' command will be:

mkinitrd -k 5.10.0-smp -f ext4 -r /dev/sda1 -n ext4 -u -o /boot/initrd.gz
root@rack:/usr/src/linux-5.10.0-rc2# k5.10.0-smp -f ext4 -r /dev/sda1 -n ext4 -u -o /boot/initrd.gz
#2999 mkdir
#boot/initrd.gz created.
Be sure to run lilo again if you use it.
root@rack:/usr/src/linux-5.10.0-rc2#
```

Rysunek 22: Stworzenie ramdisk

Dodajemy kolejny wpis oznaczający nową metodę do /etc/lilo.conf

```
root@slack:/usr/src/linux-5.18# lilo
Warning: LBA32 addressing assumed
Added Slackware_15.0 *
Added metoda_nowa +
One warning was issued.
root@slack:/usr/src/linux-5.18#
```

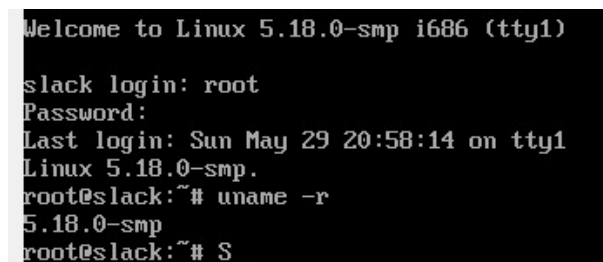
Rysunek 23: Nowe wejście w lilo.conf

Jesteśmy gotowi zrestartować maszynę i sprawdzić działanie nowego jądra. Nowy kernel pojawił się w menu wyboru:



Rysunek 24: Nowe jądro w oknie wyboru

System uruchomił się i działa poprawnie. Sprawdzamy wersję kernela aby upewnić się, że uruchomiliśmy prawidłową wersję komendą `uname -r`:



Rysunek 25: Test nowego jądra

3 Podsumowanie

Kompilacja jądra obydwoma sposobami była podobna. Zdecydowanie bardziej jednak podobała mi się kompilacja drugim, nowszym sposobem. Była odrobinę prostsza, trwała jednak kilka razy dłużej mimo tych samych ustawień maszyny wirtualnej.

Jedynym napotkanym problemem był problem z podaniem wersji kernela przy generowaniu komendy do utworzenia ramdisku. Okazało się, że przy podawaniu wersji należało podać ją

z 0 na końcu, tj. 5.18.0 zamiast samego 5.18. Był to problem stosunkowo łatwy do rozwiązania, wystarczyło spojrzeć na wynik kompilacji jądra.