

Kompilacja jądra

Paweł Nowak

10 czerwca 2022

Rozdział 1

Pobieranie ostatniej stabilnej wersji jądra

Została sprawdzona wersja jądra z użyciem komendy `uname --release`. W systemie znajdowało się jądro w wersji 5.15-27-smp. Korzystając z witryny www.kernel.org sprawdzono najnowszą stabilną wersję jądra (5.18.3).

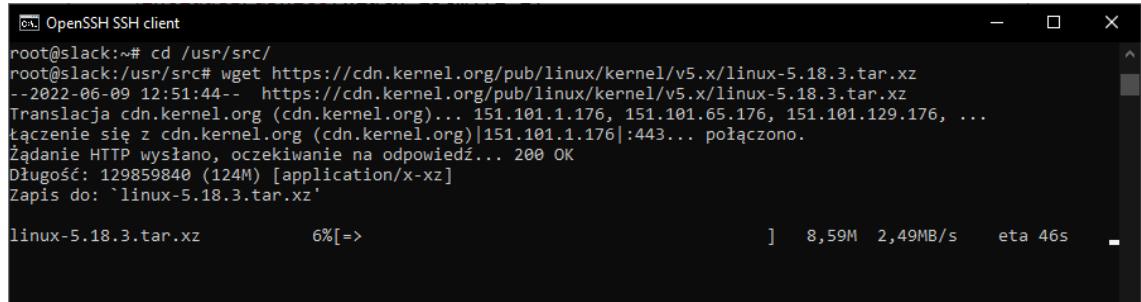
```
root@slack:~# uname --release
5.15.27-smp
root@slack:~# S_
```

Rysunek 1.1: Informacje o jądrze systemu



Rysunek 1.2: Fragment witryny www.kernel.org

Pobrano najnowszą wersję jądra, uprzednio wchodząc do katalogu `/usr/src`

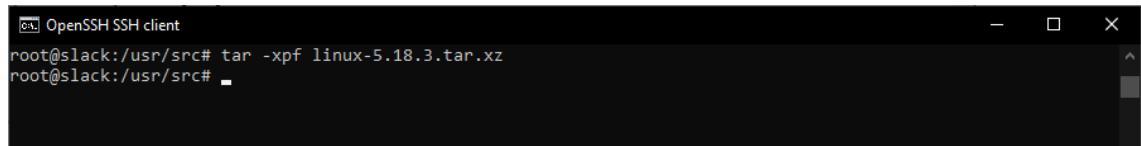


```
root@slack:~# cd /usr/src/
root@slack:/usr/src# wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.3.tar.xz
--2022-06-09 12:51:44--  https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.3.tar.xz
Translacja cdn.kernel.org (cdn.kernel.org)... 151.101.1.176, 151.101.65.176, 151.101.129.176, ...
Łączenie się z cdn.kernel.org (cdn.kernel.org)|151.101.1.176|:443... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: 129859840 (124M) [application/x-xz]
Zapis do: `linux-5.18.3.tar.xz'

linux-5.18.3.tar.xz          6%[=>]   8,59M  2,49MB/s  eta 46s
```

Rysunek 1.3: Pobieranie jądra systemu w wersji 5.18.3

Rozpakowano pobrane archiwum z użyciem komendy `tar -xpf linux-5.18.3.tar.xz`



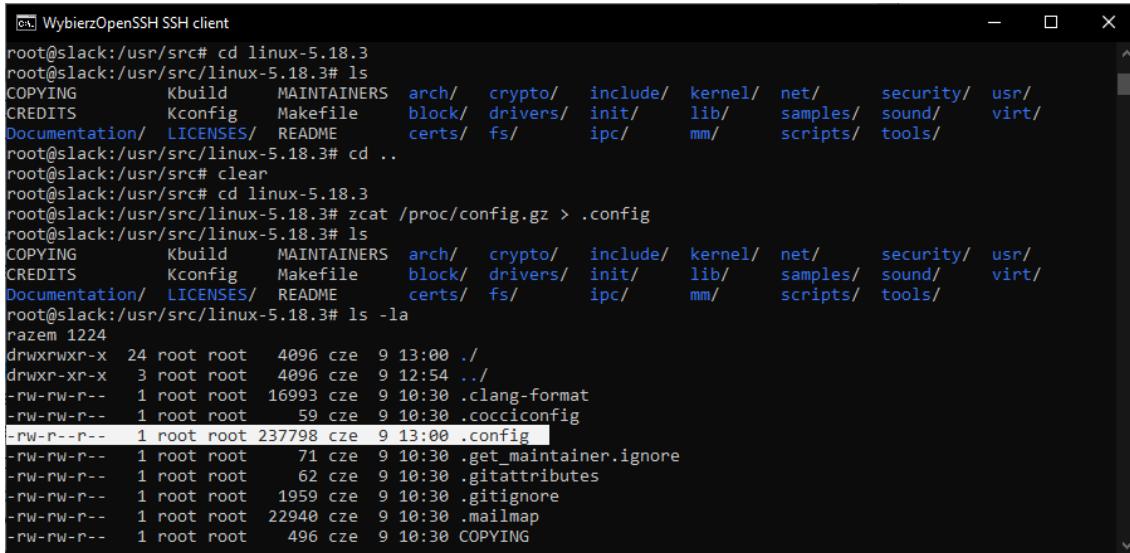
```
root@slack:/usr/src# tar -xpf linux-5.18.3.tar.xz
root@slack:/usr/src#
```

Rysunek 1.4: Rozpakowywanie archiwum z pobranym jądrem systemu

Rozdział 2

Przebieg procesu kompilacji dla starej metody

Po wejściu w katalog `linux-5.18.3` wykonano kopię konfiguracji starego jądra. W tym celu wykorzystano komendę `zcat /proc/config.gz > .config`. Rezultat komendy `ls -la` potwierdza poprawne skopiowanie pliku konfiguracyjnego (zaznaczono plik na zrzucie ekranu).



The screenshot shows a terminal window titled "WybierzOpenSSH SSH client". The terminal session is as follows:

```
root@slack:/usr/src# cd linux-5.18.3
root@slack:/usr/src/linux-5.18.3# ls
COPYING      Kbuild      MAINTAINERS  arch/  crypto/  include/  kernel/  net/      security/  usr/
CREDITS      Kconfig     Makefile    block/  drivers/  init/    lib/      samples/  sound/   virt/
Documentation/ LICENSES/ README    certs/   fs/       ipc/    mm/      scripts/  tools/
root@slack:/usr/src/linux-5.18.3# cd ..
root@slack:/usr/src# clear
root@slack:/usr/src# cd linux-5.18.3
root@slack:/usr/src/linux-5.18.3# zcat /proc/config.gz > .config
root@slack:/usr/src/linux-5.18.3# ls
COPYING      Kbuild      MAINTAINERS  arch/  crypto/  include/  kernel/  net/      security/  usr/
CREDITS      Kconfig     Makefile    block/  drivers/  init/    lib/      samples/  sound/   virt/
Documentation/ LICENSES/ README    certs/   fs/       ipc/    mm/      scripts/  tools/
root@slack:/usr/src/linux-5.18.3# ls -la
razem 1224
drwxrwxr-x  24 root root  4096 cze  9 13:00 .
drwxr-xr-x  3 root root  4096 cze  9 12:54 ..
-rw-rw-r--  1 root root 16993 cze  9 10:30 .clang-format
-rw-rw-r--  1 root root   59 cze  9 10:30 .cocciconfig
-rw-r--r--  1 root root 237798 cze  9 13:00 .config
-rw-rw-r--  1 root root   71 cze  9 10:30 .get_maintainer.ignore
-rw-rw-r--  1 root root   62 cze  9 10:30 .gitattributes
-rw-rw-r--  1 root root  1959 cze  9 10:30 .gitignore
-rw-rw-r--  1 root root 22940 cze  9 10:30 .mailmap
-rw-rw-r--  1 root root   496 cze  9 10:30 COPYING
```

Rysunek 2.1: Kopiowanie pliku konfiguracyjnego

Następnie z użyciem komendy `make localmodconfig` przygotowano plik konfiguracyjny. W każdym pojawiającym się komunikacie ustalono wartość domyślną danego parametru klikając klawisz ENTER.

```
Test the XArray code at runtime (TEST_XARRAY) [N/m/y/?] n
Perform selftest on resizable hash table (TEST_RHASHTABLE) [N/m/y/?] n
Perform selftest on siphash functions (TEST_SIPHASH) [N/m/y/?] (NEW)
Perform selftest on IDA functions (TEST_IDA) [N/m/y/?] n
Test module loading with 'hello world' module (TEST_LKM) [N/m/?] n
Test module for compilation of bitops operations (TEST_BITOPS) [N/m/?] n
Test module for stress/performance analysis of vmalloc allocator (TEST_VMALLOC) [N/m/?] n
Test user/kernel boundary protections (TEST_USER_COPY) [N/m/?] n
Test BPF filter functionality (TEST_BPF) [N/m/?] n
Test blackhole netdev functionality (TEST_BLACKHOLE_DEV) [N/m/?] n
Test find_bit functions (FIND_BIT_BENCHMARK) [N/m/y/?] n
Test firmware loading via userspace interface (TEST_FIRMWARE) [N/m/y/?] n
sysctl test driver (TEST_SYSCTL) [N/m/y/?] n
udelay test driver (TEST_UDELAY) [N/m/y/?] n
Test static keys (TEST_STATIC_KEYS) [N/m/?] n
kmod stress tester (TEST_KMOD) [N/m/?] n
Test memcat_p() helper function (TEST_MEMCAT_P) [N/m/y/?] n
Test heap/page initialization (TEST_MEMINIT) [N/m/y/?] n
Test freeing pages (TEST_FREE_PAGES) [N/m/y/?] n
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
#
root@slack:/usr/src/linux-5.18.3#
```

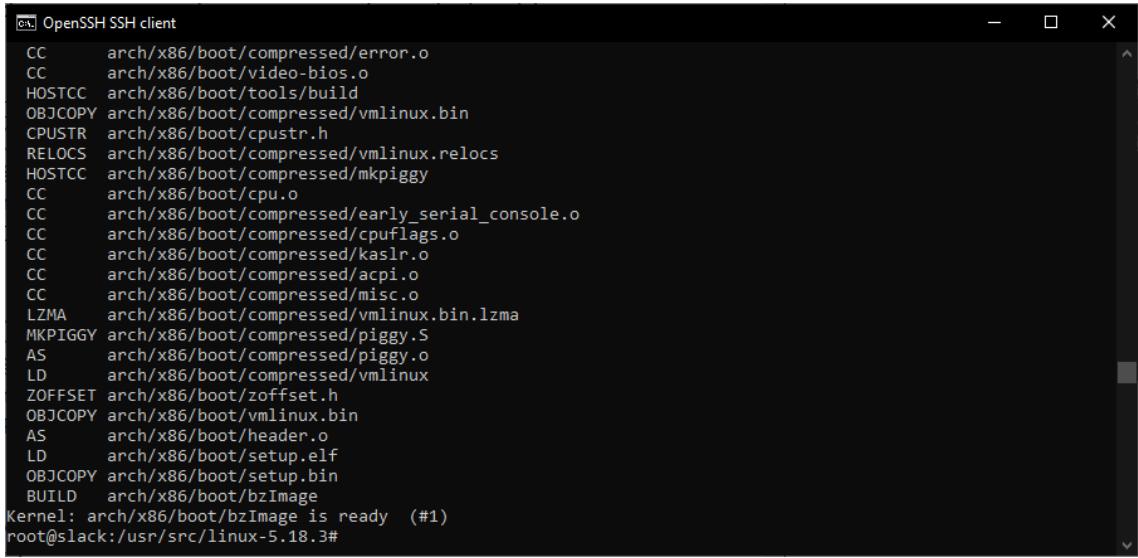
Rysunek 2.2: Rezultat komendy przygotowującej plik konfiguracyjny

Przystąpiono do procesu komplikacji jądra. Wykorzystano komendę `make -j4 bzImage`.

```
OpenSSH SSH client
root@slack:/usr/src/linux-5.18.3# make -j4 bzImage
```

Rysunek 2.3: Wywołanie komendy rozpoczynającej proces komplikacji jądra

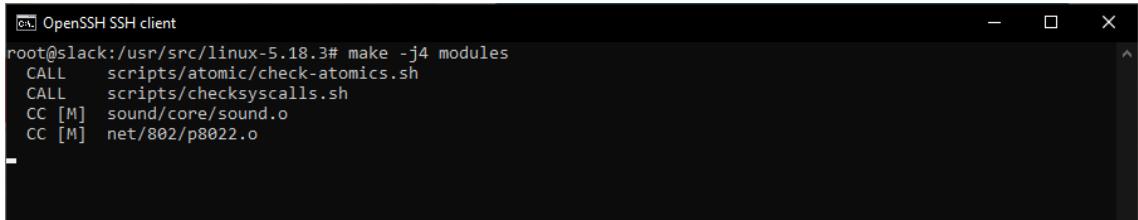
Po około 15 minutach proces komplikacji jądra zakończył się pomyślnie. Na standardowym wyjściu widnieje ścieżka do obrazu jądra: `arch/x86/boot/bzImage`.



```
CC      arch/x86/boot/compressed/error.o
CC      arch/x86/boot/video-bios.o
HOSTCC  arch/x86/boot/tools/build
OBJCOPY arch/x86/boot/compressed/vmlinux.bin
CPUSTR  arch/x86/boot/cpustr.h
RELOCS  arch/x86/boot/compressed/vmlinux.relocs
HOSTCC  arch/x86/boot/compressed/mkpiggy
CC      arch/x86/boot/cpu.o
CC      arch/x86/boot/compressed/early_serial_console.o
CC      arch/x86/boot/compressed/cpuflags.o
CC      arch/x86/boot/compressed/kaslr.o
CC      arch/x86/boot/compressed/acpi.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  (#1)
root@slack:/usr/src/linux-5.18.3#
```

Rysunek 2.4: Końcowy rezultat komendy wywołującej proces komplikacji jądra

Kolejno zbudowano moduły jądra. Wywołano komendę `make -j4 modules`.



```
root@slack:/usr/src/linux-5.18.3# make -j4 modules
CALL  scripts/atomic/check-atomics.sh
CALL  scripts/checksyscalls.sh
CC [M] sound/core/sound.o
CC [M] net/802/p8022.o
```

Rysunek 2.5: Wywołanie komendy budującej modułów jądra

```
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/sysimtblt.ko
LD [M] drivers/virt/vboxguest/vboxguest.ko
LD [M] net/802/mrp.ko
LD [M] net/802/garp.ko
LD [M] net/802/p8022.ko
LD [M] net/802/psnap.ko
LD [M] net/802/stp.ko
LD [M] net/8021q/8021q.ko
LD [M] net/ipv6/ipv6.ko
LD [M] net/l1c/l1c.ko
LD [M] net/rfkill/rfkill.ko
LD [M] net/wireless/cfg80211.ko
LD [M] sound/ac97_bus.ko
LD [M] sound/core/snd-pcm.ko
LD [M] sound/core/snd-timer.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/soundcore.ko
root@slack:/usr/src/linux-5.18.3#
```

Rysunek 2.6: Końcowy rezultat komendy budującej moduły jądra

Proces zakończył się pomyślnie po około 2 minutach. Kolejno przystąpiono do procesu instalacji modułów, co wykonano z pomocą komendy `make -j4 modules_install`.

```
root@slack:/usr/src/linux-5.18.3# make -j4 modules_install
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/acpi/ac.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/acpi/button.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/acpi/video.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/block/loop.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/char/agp/agpgart.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/char/agp/intel-gtt.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/char/agp/intel-agp.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/gpu/drm/drm.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/gpu/drm/drm_kms_helper.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/gpu/drm/drm_ttm_helper.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/gpu/drm/ttm/ttm.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/gpu/drm/vmwgfx/vmwgfx.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/i2c/algos/i2c-algo-bit.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/i2c/busses/i2c-piix4.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/i2c/i2c-core.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/input/evdev.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/input/joydev.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/input/mouse/psmouse.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/input/serio/serio_raw.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/net/ethernet/amd/pcnet32.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/net/mii.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/powercap/intel_rapl_common.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/powercap/intel_rapl_msrm.ko
INSTALL /lib/modules/5.18.3-smp/kernel/drivers/usb/host/ehci-hcd.ko
```

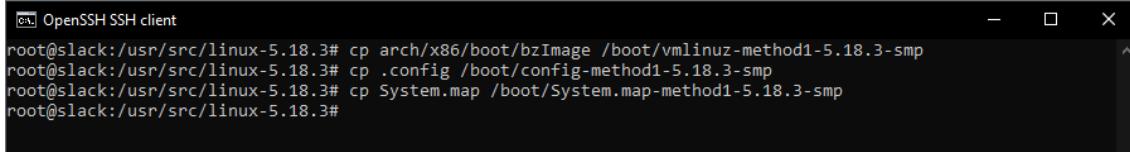
Rysunek 2.7: Instalacja modułów

Kolejnym etapem było kopiowanie plików nowego jądra do katalogu boot z użyciem poniższych komend:

- obraz jądra: `cp arch/x86/boot/bzImage boot/vmlinuz-method1-5.18.3-smp`

- plik konfiguracyjny: `cp .config /boot/config-method1-5.18.3-smp`
- tablica symboli: `cp System.map /boot/System.map-method1-5.18.3-smp`

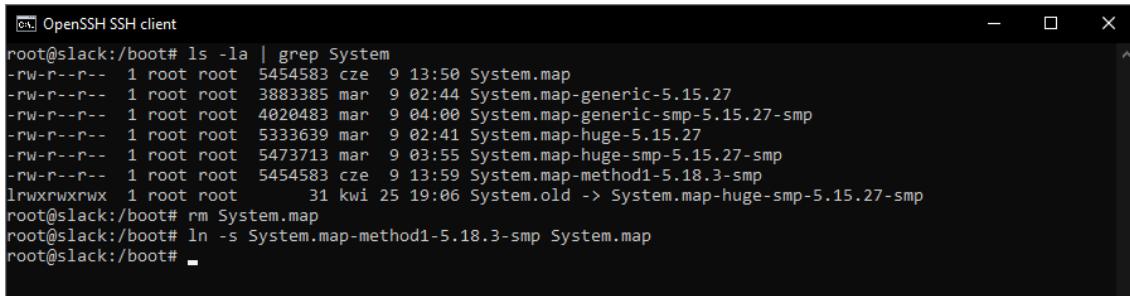
gdzie `method1` jest oznaczeniem aktualnie wykonywanej metody komplikacji jądra, a `5.18.3-smp` wersją jądra.



```
root@slack:/usr/src/linux-5.18.3# cp arch/x86/boot/bzImage /boot/vmlinuz-method1-5.18.3-smp
root@slack:/usr/src/linux-5.18.3# cp .config /boot/config-method1-5.18.3-smp
root@slack:/usr/src/linux-5.18.3# cp System.map /boot/System.map-method1-5.18.3-smp
root@slack:/usr/src/linux-5.18.3#
```

Rysunek 2.8: Kopiowanie plików jądra do katalogu boot

Kolejnym etapem jest utworzenie linku symbolicznego: do pliku `System.map` z katalogu `/boot` należy dołączyć uprzednio skopiowany plik `System.map-method1-5.18.3-smp`.

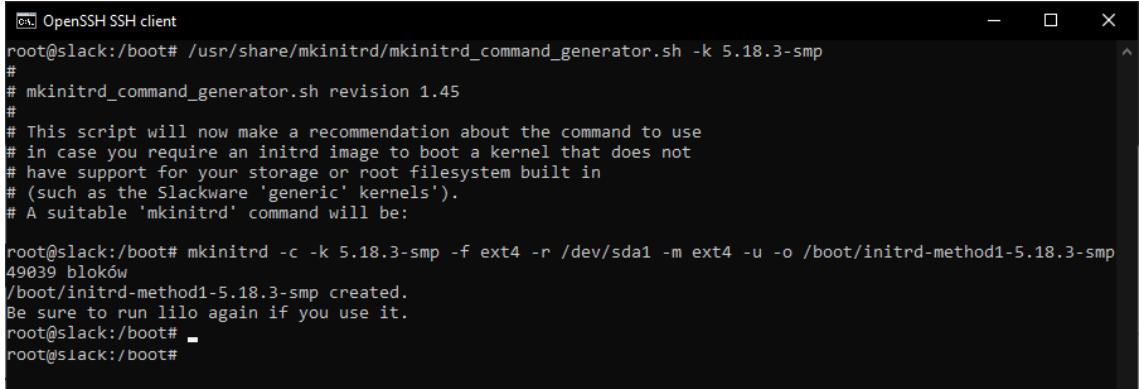


```
root@slack:/boot# ls -la | grep System
-rw-r--r-- 1 root root 5454583 cze 9 13:50 System.map
-rw-r--r-- 1 root root 3883385 mar 9 02:44 System.map-generic-5.15.27
-rw-r--r-- 1 root root 4020483 mar 9 04:00 System.map-generic-smp-5.15.27-smp
-rw-r--r-- 1 root root 5333639 mar 9 02:41 System.map-huge-5.15.27
-rw-r--r-- 1 root root 5473713 mar 9 03:55 System.map-huge-smp-5.15.27-smp
-rw-r--r-- 1 root root 5454583 cze 9 13:59 System.map-method1-5.18.3-smp
lrwxrwxrwx 1 root root      31 kwi 25 19:06 System.old -> System.map-huge-smp-5.15.27-smp
root@slack:/boot# rm System.map
root@slack:/boot# ln -s System.map-method1-5.18.3-smp System.map
root@slack:/boot#
```

Rysunek 2.9: Tworzenie symbolicznego linku do pliku `System.map`

W celu skonfigurowania RAMDISK, z użyciem skryptu `mkinitrd_command_generator.sh` wygenerowano stosowną komendę, którą później wykonano:

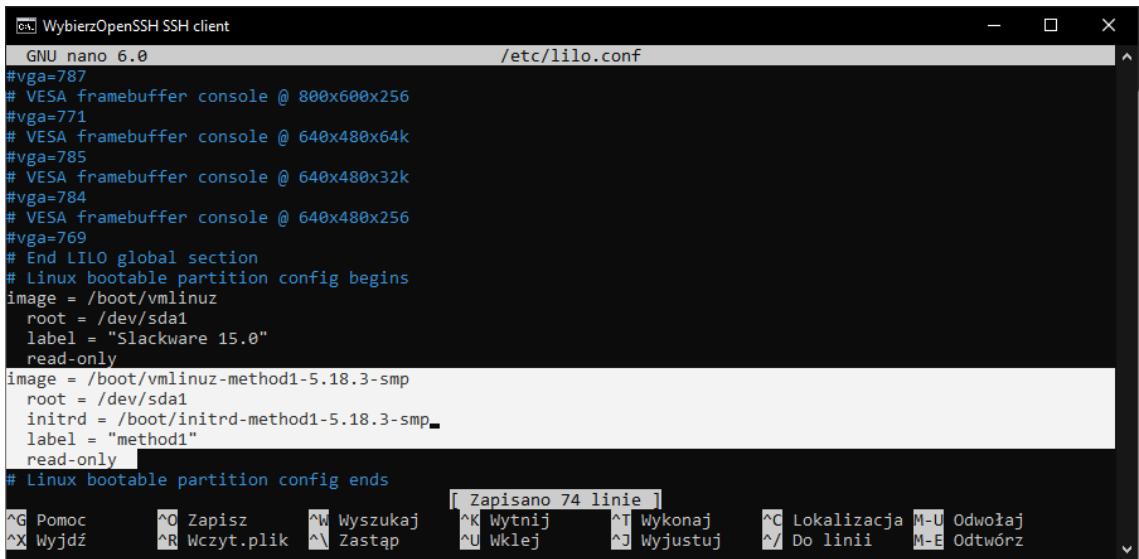
`mkinitrd -c -k 5.18.3-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz`.
W komendzie zmieniono nazwę pliku wyjściowego na taką, która zachowa spójność z poprzednio skopiowanymi plikami jądra.



```
root@slack:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.3-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:
root@slack:/boot# mkinitrd -c -k 5.18.3-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-method1-5.18.3-smp
49039 bloków
/boot/initrd-method1-5.18.3-smp created.
Be sure to run lilo again if you use it.
root@slack:/boot#
```

Rysunek 2.10: Tworzenie RAMDISK

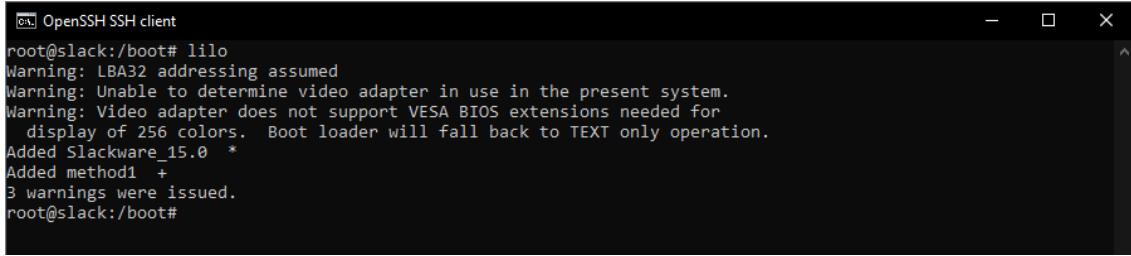
Następnie wykonano konfigurację bootloadera LILO. W tym celu edytowano plik konfiguracyjny `/etc/lilo.conf` zamieszczając w nim nowy wpis.



```
GNU nano 6.0                                     /etc/lilo.conf
#vga=787
# VESA framebuffer console @ 800x600x256
#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-5.18.3-smp
root = /dev/sda1
label = "Slackware 15.0"
read-only
image = /boot/vmlinuz-method1-5.18.3-smp
root = /dev/sda1
initrd = /boot/initrd-method1-5.18.3-smp
label = "method1"
read-only
# Linux bootable partition config ends
```

Rysunek 2.11: Edycja lilo.conf

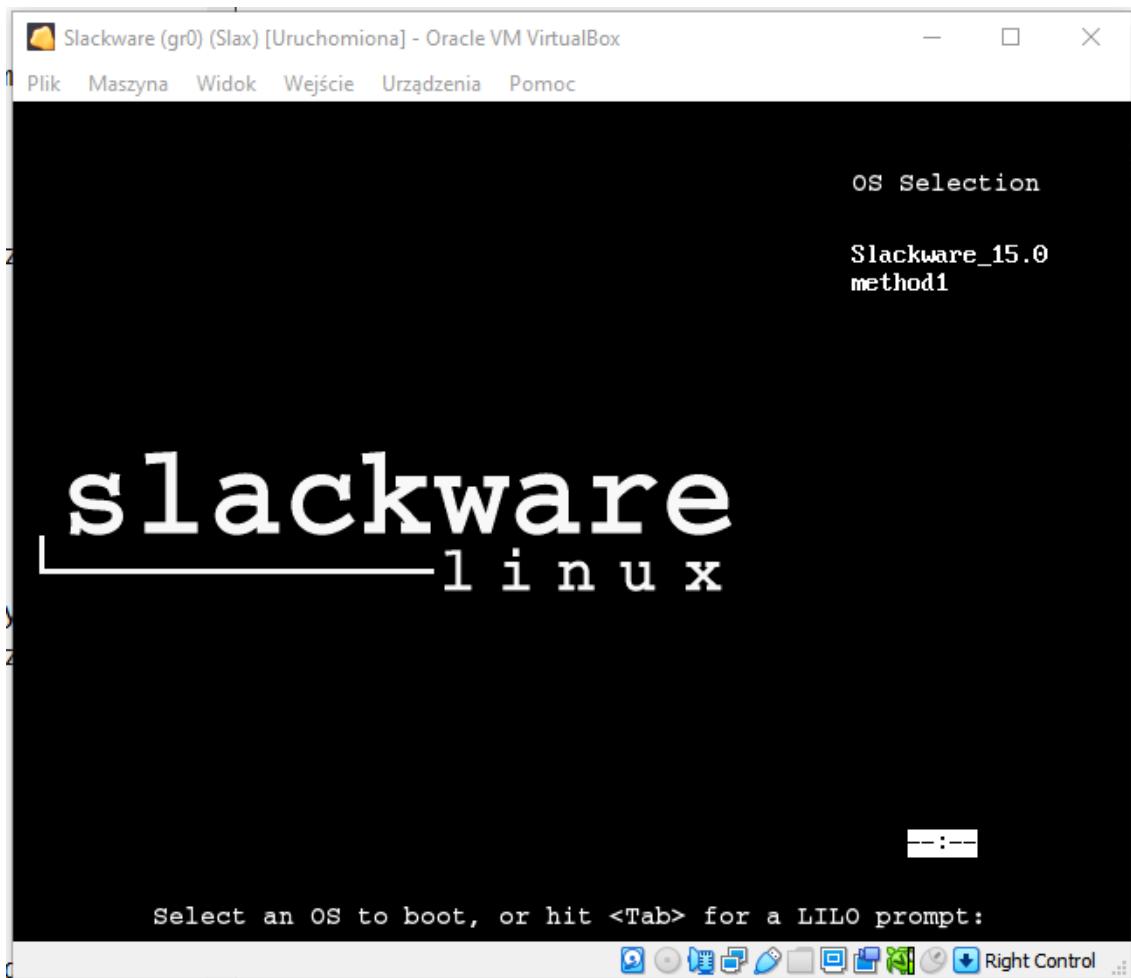
Zmiany wprowadzone do pliku konfiguracyjnego zostały zatwierdzone poprzez wywołanie komendy `lilo`:



```
OpenSSH SSH client
root@slack:/boot# lilo
Warning: LBA32 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 Slackware_15.0 *
Added method1 +
3 warnings were issued.
root@slack:/boot#
```

Rysunek 2.12: Wywołanie komendy lilo

Po zrestartowaniu systemu w boot menu pojawiła się opcja `method1`. System pomyślnie uruchomił się oraz pomyślnie zalogowano się na użytkownika `root`.



Rysunek 2.13: Boot menu po restarcie systemu

```

Slackware (gr0) (Slax) [Uruchomiona] - Oracle VM VirtualBox
Plik Maszyna Widok Wejście Urządzenia Pomoc
Maszyna wirtualna informuje, że gość zowany system operacyjny obsługuje funkcję integracji kursora myszy. Oznacza to, że nie trzeba ręcznie przechwytywać kursora myszy aby móc go używać w
eth0: adding route to 10.0.2.0/24
eth0: polling for DHCP server
dhpcd-9.4.1 starting
DUID 00:04:a9:74:57:d7:c2:cc:91:48:ad:2d:c2:58:7e:16:3d:85
eth1: waiting for carrier
eth1: carrier acquired
eth1: IAID 27:3d:3e:c4
eth1: rebinding lease of 192.168.56.101
eth1: probing address 192.168.56.101/24
eth1: leased 192.168.56.101 for 600 seconds
eth1: adding route to 192.168.56.0/24
forked to background, child pid 721
Starting system message bus: /usr/bin/dbus-uuidgen --ensure ; /usr/bin/dbus-daemon --system
Starting elogind: /lib/systemd/elogind --daemon
Starting OpenSSH SSH daemon: /usr/sbin/sshd
Starting ACPI daemon: /usr/sbin/acpid
Updating MIME database: /usr/bin/update-mime-database /usr/share/mime &
Updating gtk.immodules:
  /usr/bin/update-gtk-immodules &
Updating gdk-pixbuf.loaders:
  /usr/bin/update-gdk-pixbuf-loaders &
Compiling GSettings XML schema files:
  /usr/bin/glib-compile-schemas /usr/share/glib-2.0/schemas &
Starting crond: /usr/sbin/crond -l notice
Starting atd: /usr/sbin/atd -b 15 -1 1
Loading /usr/share/kbd/keymaps/i386/qwertyp1.map.gz
Starting gpm: /usr/sbin/gpm -f /dev/mouse -t imps2

Welcome to Linux 5.18.3-smp i686 (tty1)

slack login: root
Password:
Login incorrect

slack login: root
Password:
Login incorrect

slack login: root
Password:
Last failed login: Thu Jun  9 14:32:47 CEST 2022 on tty1
There were 2 failed login attempts since the last successful login.
Last login: Thu Jun  9 12:42:25 from 192.168.56.1
Linux 5.18.3-smp.
root@slack:~# S_

```

Rysunek 2.14: Pomyślne logowanie oraz start systemu

Po sprawdzeniu wersji jądra otrzymano rezultat 5.18.3-smp zgodny z oczekiwaniami.

```

Slackware (gr0) (Slax) [Uruchomiona] - Oracle VM VirtualBox
Plik Maszyna Widok Wejście Urządzenia Pomoc
root@slack:~# uname --release
5.18.3-smp
root@slack:~# S_

```

Rysunek 2.15: Aktualna wersja jądra

Archiwum ze zmodyfikowanymi plikami jądra zostało spakowane oraz przeniesione z użyciem następujących komend:

```

zip -r method1.zip linux-5.18.3/
scp -P 22 root@192.168.56.101:/usr/src/method1.zip .

```

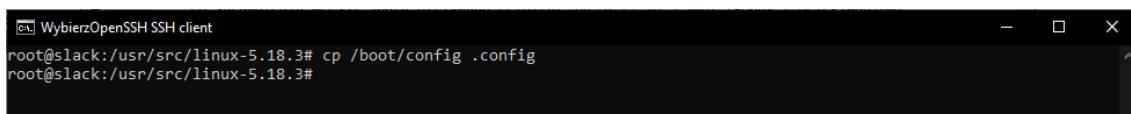
Rozdział 3

Przebieg procesu kompilacji dla nowej metody

Rozpoczęto pracę na wcześniej przygotowanym archiwum z jądrem systemu Linux. Postępowano zgodnie z zaleceniami zamieszczonymi w pliku `srcipts/kconfig/streamline_config.pl`:

```
# Howto:  
#  
# 1. Boot up the kernel that you want to streamline the config on.  
# 2. Change directory to the directory holding the source of the  
#     kernel that you just booted.  
# 3. Copy the configuration file to this directory as .config  
# 4. Have all your devices that you need modules for connected and  
#     operational (make sure that their corresponding modules are loaded)  
# 5. Run this script redirecting the output to some other file  
#     like config-strip.  
# 6. Back up your old config (if you want too).  
# 7. copy the config-strip file to .config  
# 8. Run "make oldconfig"  
#  
# Now your kernel is ready to be built with only the modules that  
# are loaded.  
#  
# Here's what I did with my Debian distribution.  
#  
# cd /usr/src/linux-2.6.10  
# cp /boot/config-2.6.10-1-686-smp .config  
# ./bin/streamline_config > config_strip  
# mv .config config.sav  
# mv config_strip .config  
# make oldconfig
```

Na początku skopiowano plik `.config` do aktualnego katalogu roboczego.

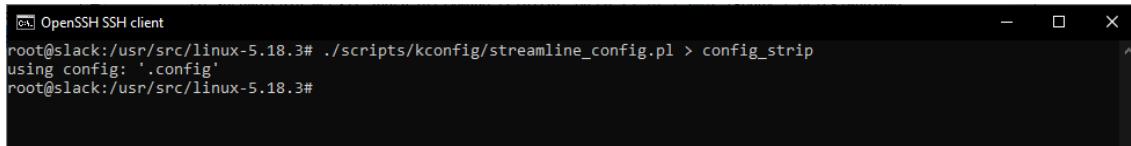


The screenshot shows a terminal window with a dark background and light-colored text. At the top, there is a header bar with the text 'WybierzOpenSSH SSH client'. The main area of the terminal shows the following command being entered:

```
root@slack:/usr/src/linux-5.18.3# cp /boot/config .config
```

Rysunek 3.1: Kopiowanie pliku `.config`

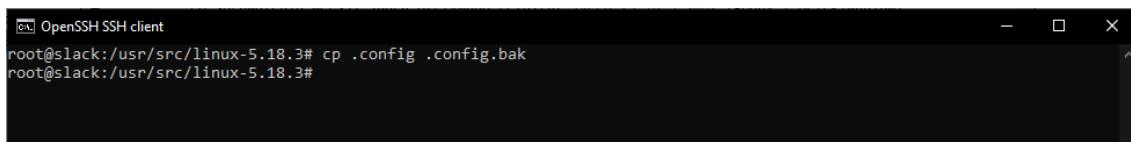
Kolejno wykonano skrypt `streamline_config.pl` przekazując go do pliku `config_strip` zgodnie z instrukcją.



```
OpenSSH SSH client
root@slack:/usr/src/linux-5.18.3# ./scripts/kconfig/streamline_config.pl > config_strip
using config: '.config'
root@slack:/usr/src/linux-5.18.3#
```

Rysunek 3.2: Wykonanie skryptu `streamline_config.pl`

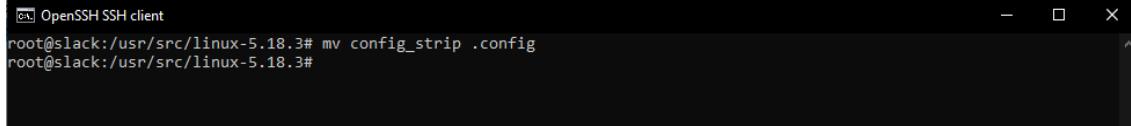
Stworzono plik `.config.bak` będący kopią zapasową pliku `.config`.



```
OpenSSH SSH client
root@slack:/usr/src/linux-5.18.3# cp .config .config.bak
root@slack:/usr/src/linux-5.18.3#
```

Rysunek 3.3: Kopiowanie pliku `.config` - kopia zapasowa

Podmieniono wcześniej utworzony plik `config_strip` z plikiem `.config`.



```
OpenSSH SSH client
root@slack:/usr/src/linux-5.18.3# mv config_strip .config
root@slack:/usr/src/linux-5.18.3#
```

Rysunek 3.4: Podmiana pliku `config_strip`

Kolejno, zgodnie z zaleceniami, wywołano komendę `make oldconfig`. W przypadku ponad 50 zapytań o parametr wskazano wartość domyślną klawiszem ENTER.



```
OpenSSH SSH client
root@slack:/usr/src/linux-5.18.3# 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
```

Rysunek 3.5: Wywołanie komendy `make oldconfig`

```
Test module for compilation of bitops operations (TEST_BITOPS) [N/m/?] n
Test module for stress/performance analysis of vmalloc allocator (TEST_VMALLOC) [N/m/?] n
Test user/kernel boundary protections (TEST_USER_COPY) [N/m/?] n
Test BPF filter functionality (TEST_BPF) [N/m/?] n
Test blackhole netdev functionality (TEST_BLACKHOLE_DEV) [N/m/?] n
Test find_bit functions (FIND_BIT_BENCHMARK) [N/m/y/?] n
Test firmware loading via userspace interface (TEST_FIRMWARE) [N/m/y/?] n
sysctl test driver (TEST_SYSCTL) [N/m/y/?] n
udelay test driver (TEST_UDELAY) [N/m/y/?] n
Test static keys (TEST_STATIC_KEYS) [N/m/?] n
kmod stress tester (TEST_KMOD) [N/m/?] n
Test memcat_p() helper function (TEST_MEMCAT_P) [N/m/y/?] n
Test heap/page initialization (TEST_MEMINIT) [N/m/y/?] n
Test freeing pages (TEST_FREE_PAGES) [N/m/y/?] n
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
#
root@slack:/usr/src/linux-5.18.3#
```

Rysunek 3.6: Rezultat końcowy komendy make oldconfig

Wykonano komendę `make bzImage` komplilującą jądro.

```
root@slack:/usr/src/linux-5.18.3# make -j4 bzImage
```

Rysunek 3.7: Wywołanie komplikacji jądra

Po niespełna 15 minutach ukończono komplikację z powodzeniem.

```
OBJCOPY arch/x86/boot/compressed/vmlinux.bin
RELOCS arch/x86/boot/compressed/vmlinux.relocs
HOSTCC arch/x86/boot/compressed/mkpiggy
CC arch/x86/boot/compressed/cpuflags.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/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@slack:/usr/src/linux-5.18.3#
```

Rysunek 3.8: Zakończenie komplikacji jądra

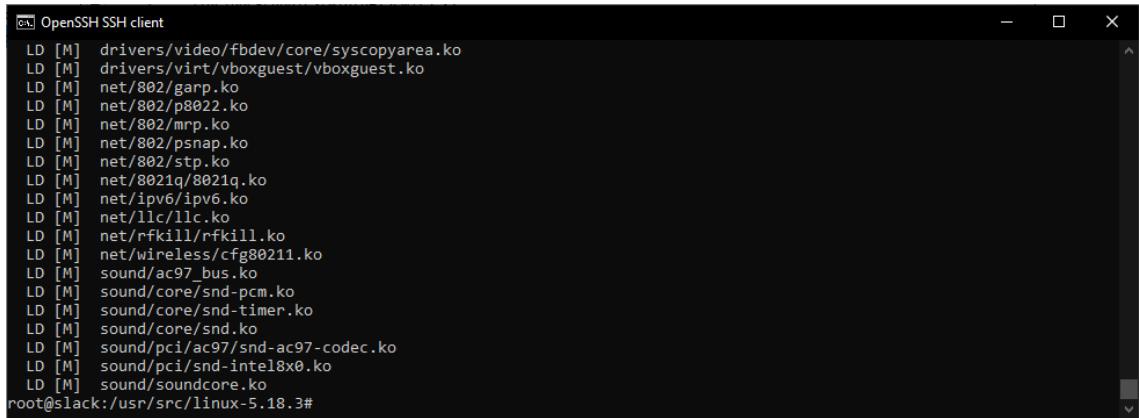
Przystąpiono do komplikacji modułów z wykorzystaniem komendy `make modules`.



```
OpenSSH SSH client
root@slack:/usr/src/linux-5.18.3# make -j4 modules
```

Rysunek 3.9: Wywołanie kompilacji modułów

Po około 2 minutach kompilacja modułów zakończyła się powodzeniem.



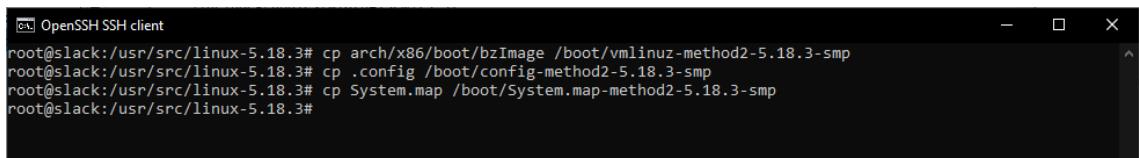
```
OpenSSH SSH client
LD [M] drivers/video/fbdev/core/syscopyarea.ko
LD [M] drivers/virt/vboxguest/vboxguest.ko
LD [M] net/802/garp.ko
LD [M] net/802/p8022.ko
LD [M] net/802/mrp.ko
LD [M] net/802/psnap.ko
LD [M] net/802/stp.ko
LD [M] net/8021q/8021q.ko
LD [M] net/ipv6/ipv6.ko
LD [M] net/l1c/l1c.ko
LD [M] net/rfkill/rfkill.ko
LD [M] net/wireless/cfg80211.ko
LD [M] sound/ac97_bus.ko
LD [M] sound/core/snd-pcm.ko
LD [M] sound/core/snd-timer.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/soundcore.ko
root@slack:/usr/src/linux-5.18.3#
```

Rysunek 3.10: Zakończenie kompilacji modułów

Kolejnym etapem było kopiowanie plików nowego jądra do katalogu boot z użyciem poniższych komend:

- obraz jądra: `cp arch/x86/boot/bzImage /boot/vmlinuz-method2-5.18.3-smp`
- plik konfiguracyjny: `cp .config /boot/config-method2-5.18.3-smp`
- tablica symboli: `cp System.map /boot/System.map-method2-5.18.3-smp`

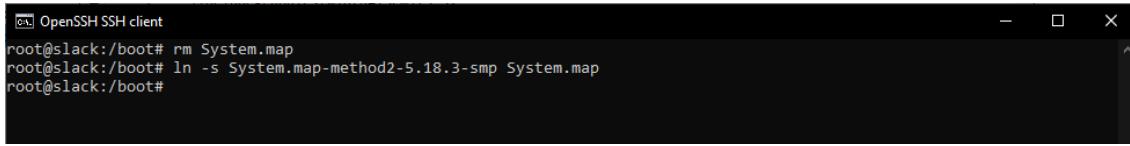
gdzie `method2` jest oznaczeniem aktualnie wykonywanej metody kompilacji jądra, a `5.18.3-smp` wersją jądra.



```
OpenSSH SSH client
root@slack:/usr/src/linux-5.18.3# cp arch/x86/boot/bzImage /boot/vmlinuz-method2-5.18.3-smp
root@slack:/usr/src/linux-5.18.3# cp .config /boot/config-method2-5.18.3-smp
root@slack:/usr/src/linux-5.18.3# cp System.map /boot/System.map-method2-5.18.3-smp
root@slack:/usr/src/linux-5.18.3#
```

Rysunek 3.11: Kopiowanie plików jądra do katalogu /boot

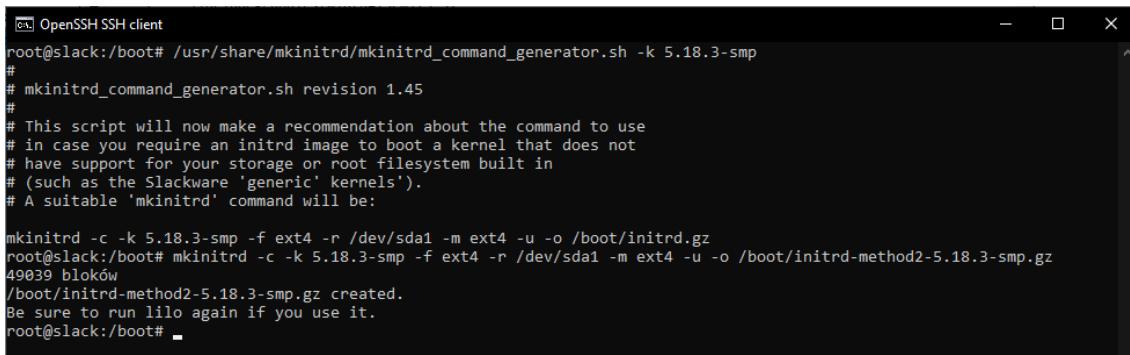
Kolejno, po otworzeniu katalogu `\boot`, skasowano stary plik `System.map` oraz utworzono odpowiedni link symboliczny.



```
OpenSSH SSH client
root@slack:/boot# rm System.map
root@slack:/boot# ln -s System.map-method2-5.18.3-smp System.map
root@slack:/boot#
```

Rysunek 3.12: Tworzenie linku symbolicznego do System.map

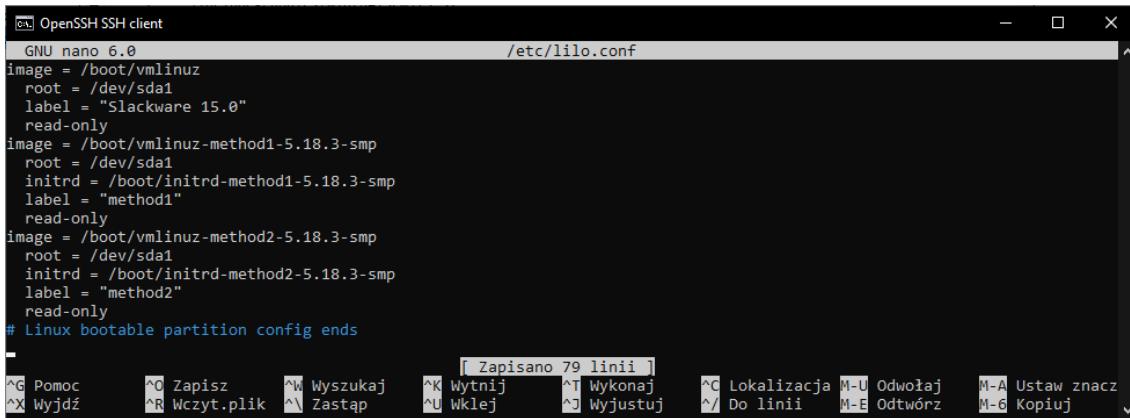
Następnie wygenerowano stosowną komendę do utworzenia RAMDISK, którą wykonano modyfikując nazwę pliku na taką, która zachowa spójność ze wcześniejszymi plikami - dodano nazwę metody oraz wersję jądra.



```
OpenSSH SSH client
root@slack:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.3-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 5.18.3-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
root@slack:/boot# mkinitrd -c -k 5.18.3-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-method2-5.18.3-smp.gz
49039 bloków
/boot/initrd-method2-5.18.3-smp.gz created.
Be sure to run lilo again if you use it.
root@slack:/boot#
```

Rysunek 3.13: Tworzenie RAMDISK

Kolejnym etapem było dodanie wpisu do pliku `/etc/lilo.conf`.

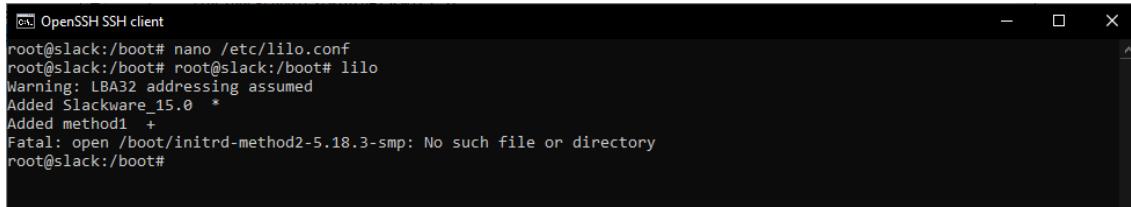


```
OpenSSH SSH client
GNU nano 6.0                                     /etc/lilo.conf
image = /boot/vmlinuz
root = /dev/sda1
label = "Slackware 15.0"
read-only
image = /boot/vmlinuz-method1-5.18.3-smp
root = /dev/sda1
initrd = /boot/initrd-method1-5.18.3-smp
label = "method1"
read-only
image = /boot/vmlinuz-method2-5.18.3-smp
root = /dev/sda1
initrd = /boot/initrd-method2-5.18.3-smp
label = "method2"
read-only
# Linux bootable partition config ends

[Zapisano 79 linii]
```

Rysunek 3.14: Edycja lilo.conf

Po zmodyfikowania wcześniej wspomnianego pliku, zatwierdzono zmiany wywołując komendę `lilo`.



```
OpenSSH SSH client
root@slack:/boot# nano /etc/lilo.conf
root@slack:/boot# root@slack:/boot# lilo
Warning: LBA32 addressing assumed
Added Slackware_15.0 *
Added method1 +
Fatal: open /boot/initrd-method2-5.18.3-smp: No such file or directory
root@slack:/boot#
```

Rysunek 3.15: Zatwierdzenie zmian w LILO

Przy rebootowaniu systemu wbrew oczekiwaniom nie ukazała się dostępna opcja `method2`. Po krótkiej analizie dotychczasowej dokumentacji (zrzutów ekranu) udało się zidentyfikować błąd: na zrzucie 3.15 na standardowym wyjściu pojawiła się informacja o braku pliku `/boot/initrd-method2-5.18.3-smp`. W celu szerszego rozpoznania błędu i naprawy wykonoano następujące czynności:

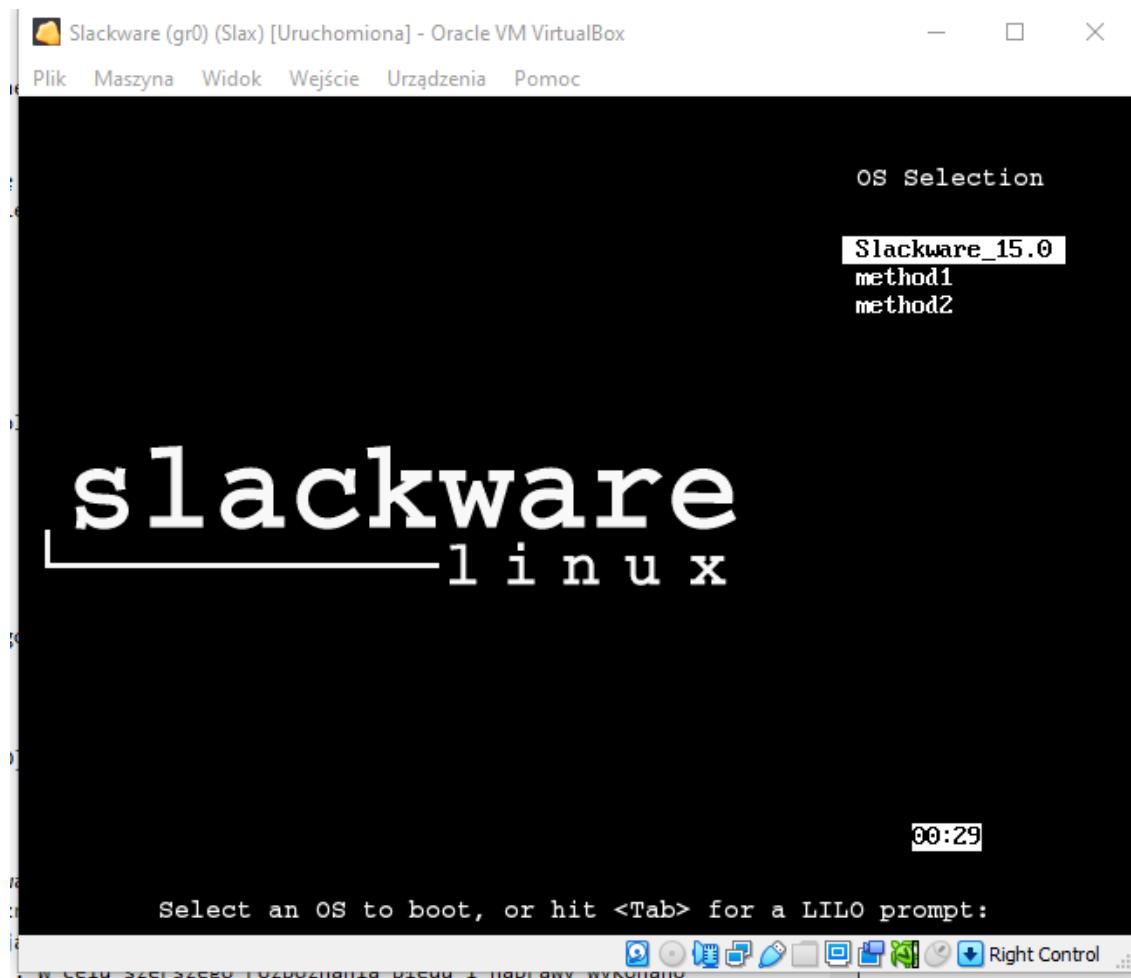
- Sprawdzono, czy prawidłowa ścieżka znajduje się w pliku `lilo.conf`
- Wylistowano katalog `/boot` - zauważono błędą nazwę pliku `initrd`
- Zmieniono nazwę pliku `initrd` na poprawną
- Powtórnie wykonano komendę `lilo`

```
#uga=773
# VESA framebuffer console @ 800x600x64k
#uga=788
# VESA framebuffer console @ 800x600x32k
#uga=787
# VESA framebuffer console @ 800x600x256
#uga=771
# VESA framebuffer console @ 640x480x64k
#uga=785
# VESA framebuffer console @ 640x480x32k
#uga=784
# VESA framebuffer console @ 640x480x256
#uga=769
# End LILO global section
# Linux bootable partition config begins
image = /boot/vmlinuz
root = /dev/sda1
label = "Slackware 15.0"
read-only
image = /boot/vmlinuz-method1-5.18.3-smp
root = /dev/sda1
initrd = /boot/initrd-method1-5.18.3-smp
label = "method1"
read-only
image = /boot/vmlinuz-method2-5.18.3-smp
root = /dev/sda1
initrd = /boot/initrd-method2-5.18.3-smp
label = "method2"
read-only
# Linux bootable partition config ends

root@slack:~# cd /boot
root@slack:/boot# ls | grep initrd
README.initrd
initrd-method1-5.18.3-smp
initrd-method2-5.18.3-smp.gz
initrd-tree/
initrd.gz
root@slack:/boot# mv initrd-method2-5.18.3-smp.gz initrd-method2-5.18.3-smp
root@slack:/boot# lilo
Warning: LBA32 addressing assumed
Added Slackware_15.0 *
Added method1 +
Added method2 +
One warning was issued.
root@slack:/boot# S
```

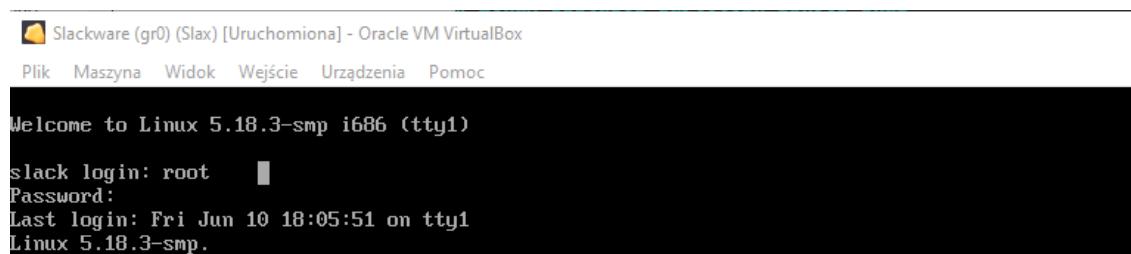
Rysunek 3.16: Naprawa błędu

Po poprawieniu błędu przystąpiono ponownie do rebootu systemu. Tym razem w boot menu ukazała się etykieta `method2`.



Rysunek 3.17: Boot menu

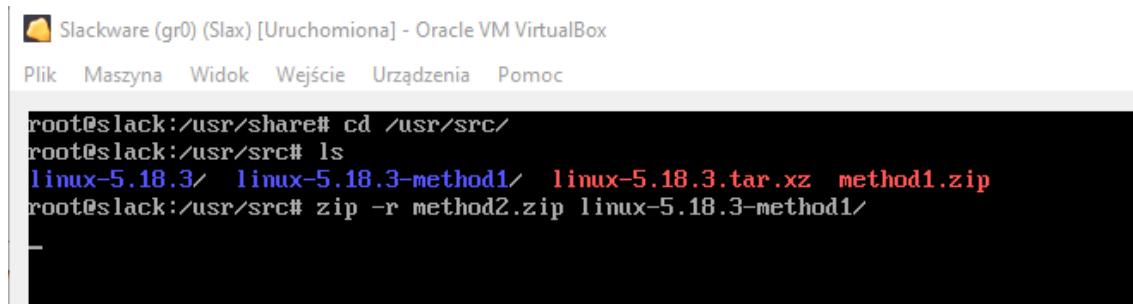
Udało się pomyślnie zalogować do systemu oraz została wyświetlona informacja o wersji jądra.



Rysunek 3.18: Pomyślne logowanie do systemu

Stworzono archiwum `method2.zip` zawierające pliki jądra dla drugiej metody. Pliki zo-

stały skopiowane do innego systemu analogicznie do sposobu zaprezentowanego w rozdziale z pierwszą metodą kompilacji jądra.



The screenshot shows a terminal window titled "Slackware (gr0) (Slax) [Uruchomiona] - Oracle VM VirtualBox". The window contains a command-line session:

```
root@slack:/usr/share# cd /usr/src/
root@slack:/usr/src# ls
linux-5.18.3/ linux-5.18.3-method1/ linux-5.18.3.tar.xz method1.zip
root@slack:/usr/src# zip -r method2.zip linux-5.18.3-method1/
[...]
```

Rysunek 3.19: Tworzenie archiwum

Rozdział 4

Podsumowanie i wnioski końcowe

Podczas komplikacji jądra obiema metodami nie napotkałem większych trudności. Jedyny błąd przytrafił się podczas drugiej metody i wynikał tylko z mojego niedopatrzenia, a wystąpił podczas kopiowania plików - podałem błędną nazwę. Oba rozwiązania wydają się dość podobne, mają bardzo dużą część wspólną. Bardzo pomocna okazała się instrukcja zawierająca konkretne przykłady komend w przypadku metody drugiej.