

Kompilacja jądra Linuxa

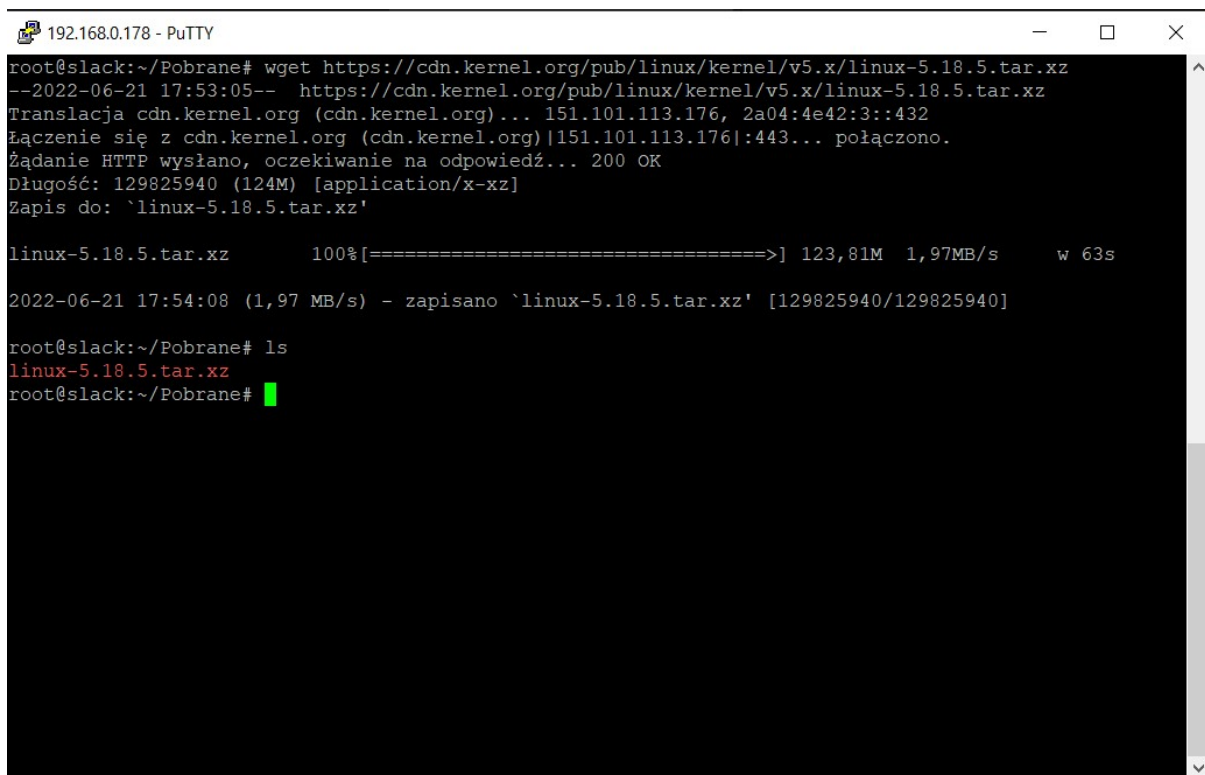
Paweł Lysko

22 czerwca 2022

Rozdział 1

Pobranie jądra

Ze względu na używanie innego komputera do hostowania wirtualnej maszyny, używam programu **PuTTY** do połączenia się przez ssh. W momencie pobierania, najnowszą dostępną wersją jądra to 5.18.5. Pobieram ją za pomocą polecenia `wget`.



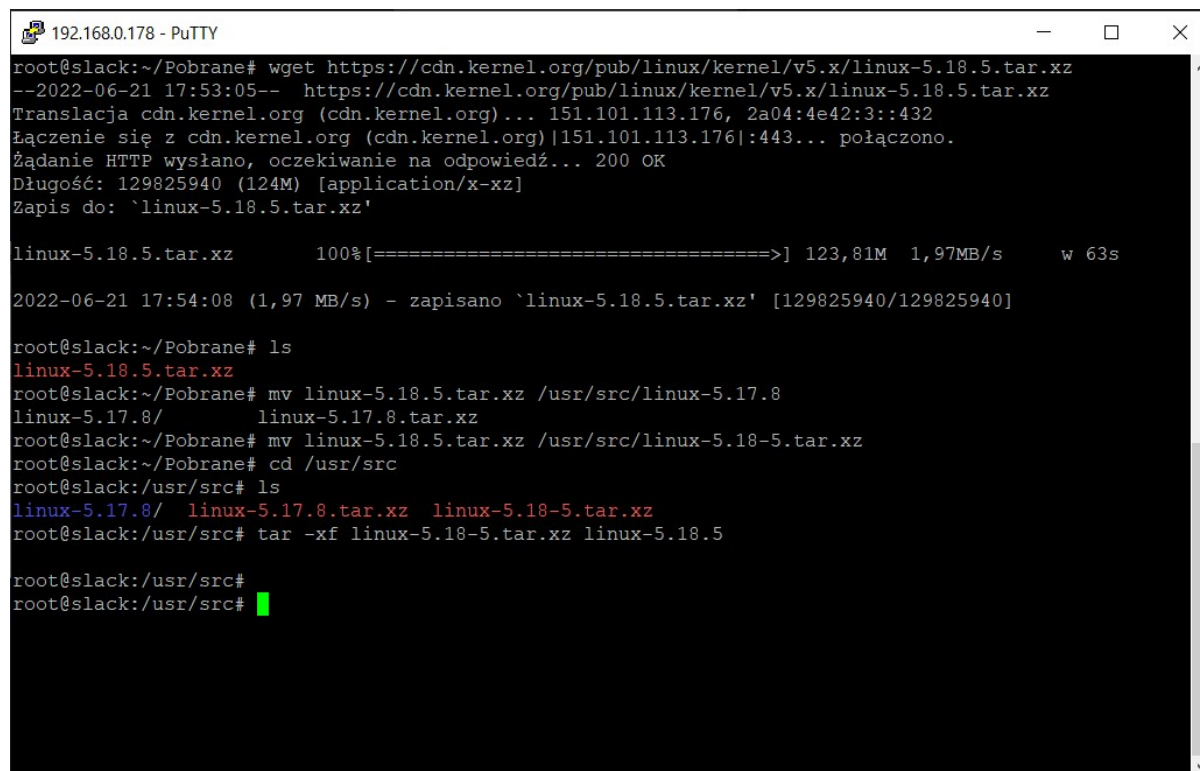
```
192.168.0.178 - PuTTY
root@slack:~/Pobrane# wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.5.tar.xz
--2022-06-21 17:53:05-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.5.tar.xz
Translacja cdn.kernel.org (cdn.kernel.org)... 151.101.113.176, 2a04:4e42:3::432
Łączenie się z cdn.kernel.org (cdn.kernel.org)[151.101.113.176]:443... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: 129825940 (124M) [application/x-xz]
Zapis do: `linux-5.18.5.tar.xz'

linux-5.18.5.tar.xz      100%[=====>] 123,81M  1,97MB/s      w 63s
2022-06-21 17:54:08 (1,97 MB/s) - zapisano `linux-5.18.5.tar.xz' [129825940/129825940]

root@slack:~/Pobrane# ls
linux-5.18.5.tar.xz
root@slack:~/Pobrane#
```

Rysunek 1.1: Pobranie najnowszej wersji kernela

Do rozpakowania archiwum używam polecenia **tar -xf linux-5.18.5.tar.xz linux-5.18.5**



```
192.168.0.178 - PuTTY
root@slack:~/Pobrane# wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.5.tar.xz
--2022-06-21 17:53:05-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.5.tar.xz
Translacja cdn.kernel.org (cdn.kernel.org)... 151.101.113.176, 2a04:4e42:3::432
Łączenie się z cdn.kernel.org (cdn.kernel.org)[151.101.113.176]:443... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: 129825940 (124M) [application/x-xz]
Zapis do: `linux-5.18.5.tar.xz'

linux-5.18.5.tar.xz      100%[=====>] 123,81M  1,97MB/s    w 63s
2022-06-21 17:54:08 (1,97 MB/s) - zapisano `linux-5.18.5.tar.xz' [129825940/129825940]

root@slack:~/Pobrane# ls
linux-5.18.5.tar.xz
root@slack:~/Pobrane# mv linux-5.18.5.tar.xz /usr/src/linux-5.17.8
linux-5.17.8/          linux-5.17.8.tar.xz
root@slack:~/Pobrane# mv linux-5.18.5.tar.xz /usr/src/linux-5.18-5.tar.xz
root@slack:~/Pobrane# cd /usr/src
root@slack:/usr/src# ls
linux-5.17.8/  linux-5.17.8.tar.xz  linux-5.18-5.tar.xz
root@slack:/usr/src# tar -xf linux-5.18-5.tar.xz linux-5.18.5

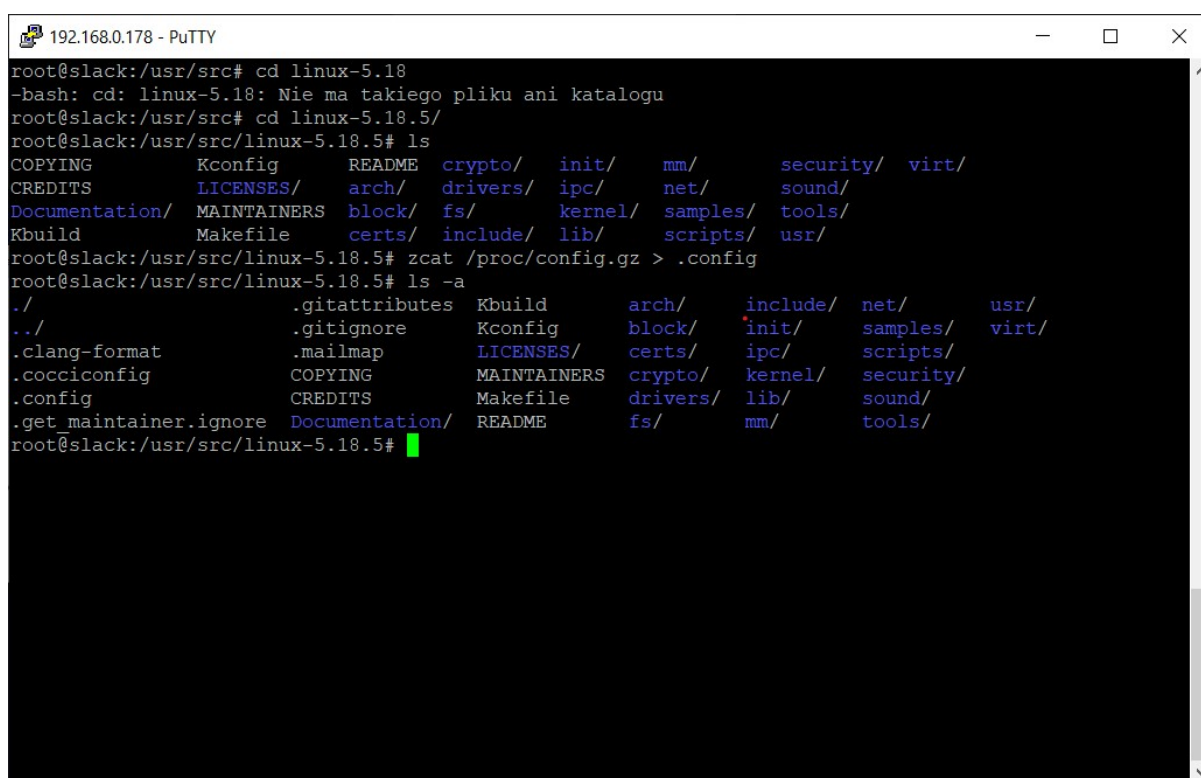
root@slack:/usr/src#
root@slack:/usr/src#
```

Rysunek 1.2: Rozpakowanie archiwum

Rozdział 2

Kompilacja jądra z wykorzystaniem starej metody make localconfig

Na początku skopiowałem obecny plik konfiguracyjny(Rys. 2.1).



```
192.168.0.178 - PuTTY
root@slack:/usr/src# cd linux-5.18
-bash: cd: linux-5.18: Nie ma takiego pliku ani katalogu
root@slack:/usr/src# cd linux-5.18.5/
root@slack:/usr/src/linux-5.18.5# ls
COPYING      Kconfig      README      crypto/     init/       mm/         security/   virt/
CREDITS      LICENSES/    arch/       drivers/    ipc/        net/        sound/
Documentation/ MAINTAINERS  block/      fs/         kernel/     samples/    tools/
Kbuild       Makefile     certs/      include/    lib/        scripts/    usr/
root@slack:/usr/src/linux-5.18.5# zcat /proc/config.gz > .config
root@slack:/usr/src/linux-5.18.5# ls -a
./          .gitattributes Kbuild      arch/       include/    net/        usr/
../         .gitignore    Kconfig     block/      init/       samples/    virt/
.clang-format .mailmap     LICENSES/   certs/      ipc/        scripts/
.cocciconfig  COPYING      MAINTAINERS crypto/      kernel/     security/
.config       CREDITS      Makefile    drivers/    lib/        sound/
.get_maintainer.ignore Documentation/ README       fs/         mm/         tools/
root@slack:/usr/src/linux-5.18.5#
```

Rysunek 2.1: Skopiowanie aktualnej konfiguracji

Do utworzenia nowej konfiguracji używam polecenia *make localmodconfig*(Ry. 2.2). Wszystkie ustawienia pozostawiłem domyślnie (Rys.2.3)

```

192.168.0.178 - PuTTY
./      .gitattributes  Kbuild      arch/       include/    net/        usr/
./      .gitignore       Kconfig     block/      init/       samples/    virt/
.clang-format  .mailmap      LICENSES/   certs/      ipc/        scripts/
.cocciconfig   COPYING       MAINTAINERS crypto/      kernel/     security/
.config        CREDITS       Makefile    drivers/    lib/        sound/
.get_maintainer.ignore Documentation/ README       fs/         mm/         tools/
root@slack:/usr/src/linux-5.18.5# make localmodconfig
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
using config: '.config'
*
* Restart config...
*
*
* Timers subsystem
*
Timer tick handling
  1. Periodic timer ticks (constant rate, no dynticks) (HZ_PERIODIC)
> 2. Idle dynticks system (tickless idle) (NO_HZ_IDLE)
choice[1-2]: 2

```

Rysunek 2.2: wykonanie polecenia make localmodconfig

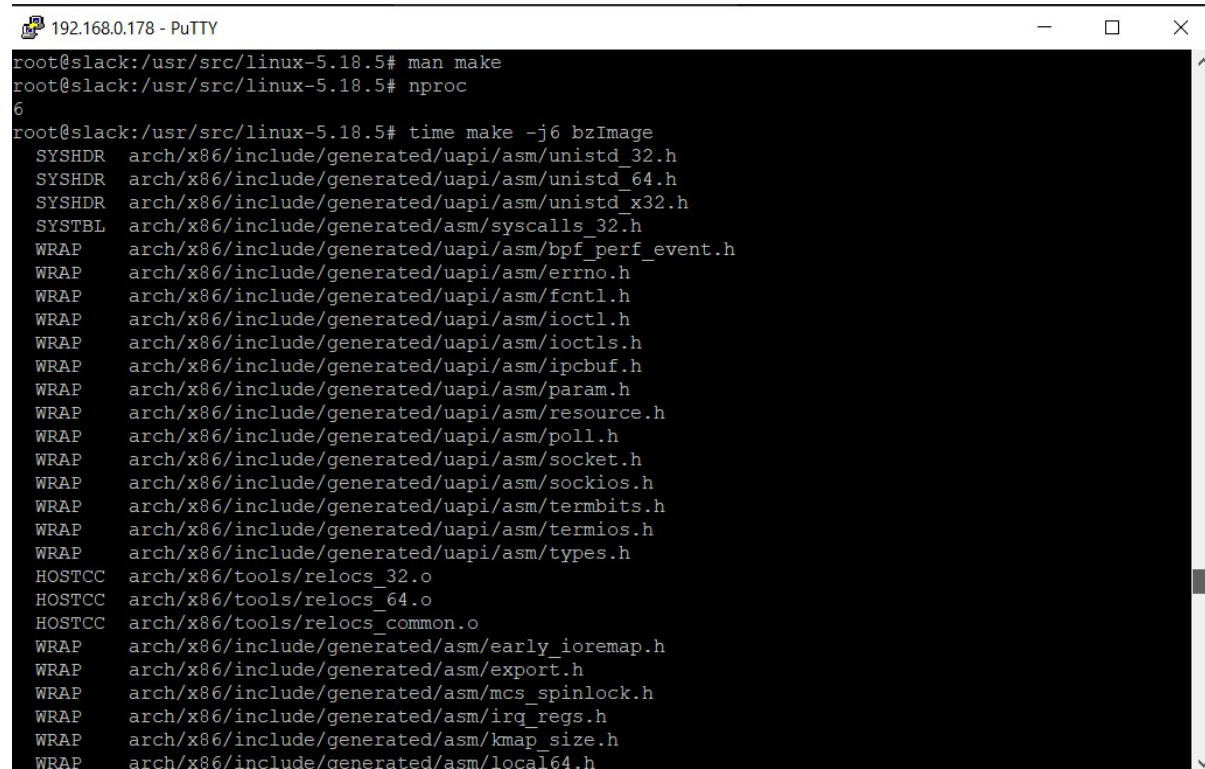
```

192.168.0.178 - PuTTY
Test scanf() family of functions at runtime (TEST_SCANF) [N/m/y/?] n
Test bitmap_*() family of functions at runtime (TEST_BITMAP) [N/m/y/?] n
Test functions located in the uuid module at runtime (TEST_UUID) [N/m/y/?] n
Test the XArray code at runtime (TEST_XARRAY) [N/m/y/?] n
Perform selftest on resizable hash table (TEST_RHASH_TABLE) [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-5.18.5#
root@slack:/usr/src/linux-5.18.5#
root@slack:/usr/src/linux-5.18.5#
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 2.3: Akceptacja domyślnych ustawień

Po utworzeniu konfiguracji przystąpiłem do kompilacji jądra. Za pomocą polecenia **nproc** liczbę dostępnych procesorów. Do kompilacji użyłem polecenia *time make -j6 bzImage*. Żeby kompilacja była jak najszybsza do argumentu -j użyłem liczby otrzymanej z polecenia *nproc*. Komendy *time* użyłem w celu zmierzenia czasu kompilacji(Rys. 2.4).



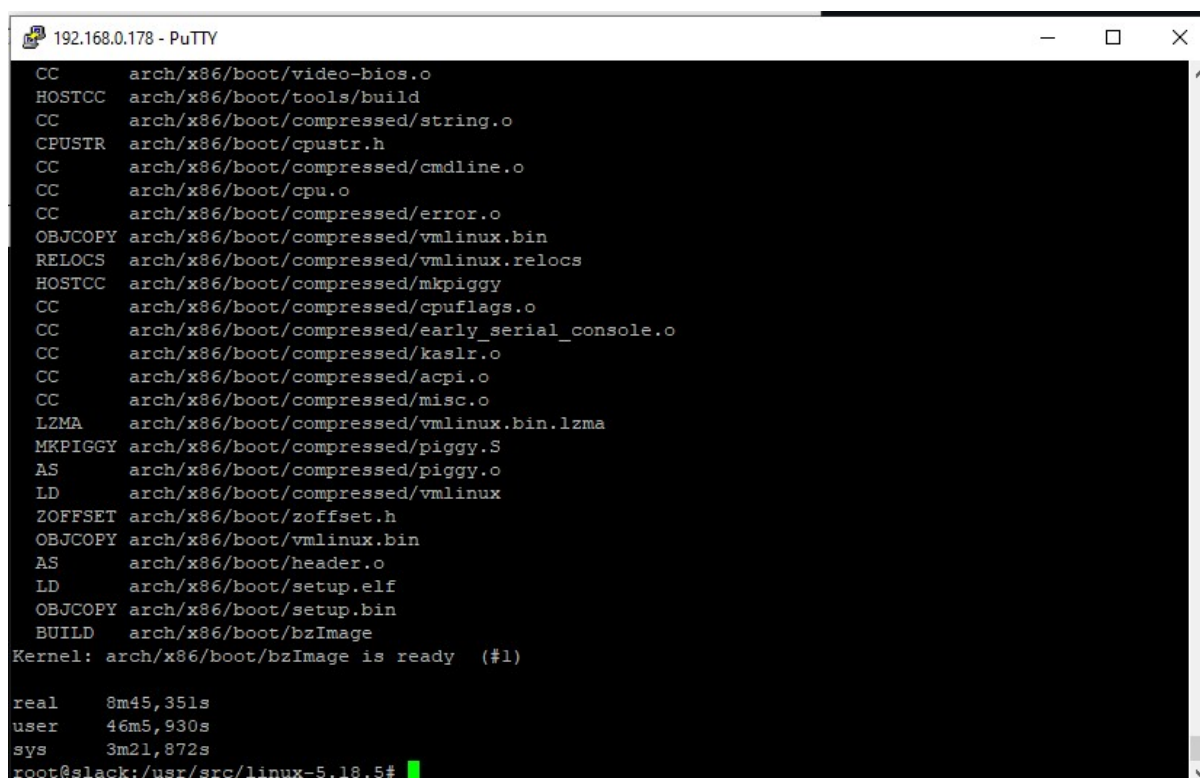
```

192.168.0.178 - PuTTY
root@slack:/usr/src/linux-5.18.5# man make
root@slack:/usr/src/linux-5.18.5# nproc
6
root@slack:/usr/src/linux-5.18.5# time 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
WRAP arch/x86/include/generated/uapi/asm/ipcbuf.h
WRAP arch/x86/include/generated/uapi/asm/param.h
WRAP arch/x86/include/generated/uapi/asm/resource.h
WRAP arch/x86/include/generated/uapi/asm/poll.h
WRAP arch/x86/include/generated/uapi/asm/socket.h
WRAP arch/x86/include/generated/uapi/asm/sockios.h
WRAP arch/x86/include/generated/uapi/asm/termbits.h
WRAP arch/x86/include/generated/uapi/asm/termios.h
WRAP arch/x86/include/generated/uapi/asm/types.h
HOSTCC arch/x86/tools/relocs_32.o
HOSTCC arch/x86/tools/relocs_64.o
HOSTCC arch/x86/tools/relocs_common.o
WRAP arch/x86/include/generated/asm/early_ioremap.h
WRAP arch/x86/include/generated/asm/export.h
WRAP arch/x86/include/generated/asm/mcs_spinlock.h
WRAP arch/x86/include/generated/asm/irq_regs.h
WRAP arch/x86/include/generated/asm/kmap_size.h
WRAP arch/x86/include/generated/asm/local64.h

```

Rysunek 2.4: Pierwsza kompilacja jądra z komendą *time*

Kompilacja trwała 8 minut i 46 sekund.(Rys. 2.5). Po kompilacji przystąpiłem do budowania modułów. Użyłem do tego polecenia *time make -j6 modules*(Rys. 2.6). Budowanie trwało 54 sekundy(Rys. 2.7). Po ich zbudowaniu użyłem polecenia *make -j6 modules_install* do ich zainstalowania(Rys. 2.8).



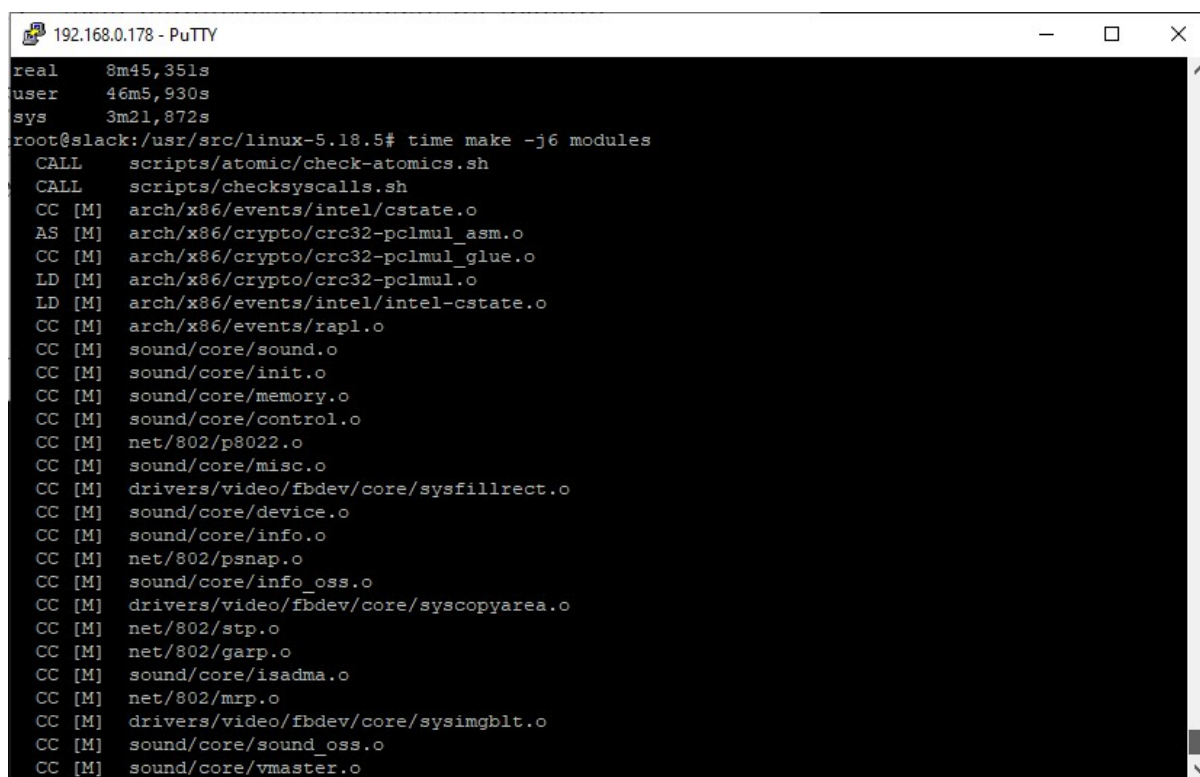
```

192.168.0.178 - PuTTY
CC      arch/x86/boot/video-bios.o
HOSTCC  arch/x86/boot/tools/build
CC      arch/x86/boot/compressed/string.o
CPUSTR  arch/x86/boot/cpustr.h
CC      arch/x86/boot/compressed/cmdline.o
CC      arch/x86/boot/cpu.o
CC      arch/x86/boot/compressed/error.o
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)

real    8m45,351s
user    46m5,930s
sys     3m21,872s
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 2.5: Zakończenie pierwszej kompilacji



```

192.168.0.178 - PuTTY
real    8m45,351s
user    46m5,930s
sys     3m21,872s
root@slack:/usr/src/linux-5.18.5# time make -j6 modules
CALL    scripts/atomic/check-atomics.sh
CALL    scripts/checksyscalls.sh
CC [M]  arch/x86/events/intel/cstate.o
AS [M]  arch/x86/crypto/crc32-pclmul_asm.o
CC [M]  arch/x86/crypto/crc32-pclmul_glue.o
LD [M]  arch/x86/crypto/crc32-pclmul.o
LD [M]  arch/x86/events/intel/intel-cstate.o
CC [M]  arch/x86/events/rapl.o
CC [M]  sound/core/sound.o
CC [M]  sound/core/init.o
CC [M]  sound/core/memory.o
CC [M]  sound/core/control.o
CC [M]  net/802/p8022.o
CC [M]  sound/core/misc.o
CC [M]  drivers/video/fbdev/core/sysfillrect.o
CC [M]  sound/core/device.o
CC [M]  sound/core/info.o
CC [M]  net/802/psnap.o
CC [M]  sound/core/info_oss.o
CC [M]  drivers/video/fbdev/core/syscopyarea.o
CC [M]  net/802/stp.o
CC [M]  net/802/garp.o
CC [M]  sound/core/isadma.o
CC [M]  net/802/mrp.o
CC [M]  drivers/video/fbdev/core/sysimgblt.o
CC [M]  sound/core/sound_oss.o
CC [M]  sound/core/vmaster.o

```

Rysunek 2.6: Pierwsze budowanie modułów.

```

LD [M] drivers/net/ethernet/amd/pcnet32.ko
LD [M] drivers/net/mii.ko
LD [M] drivers/powercap/intel_rapl_common.ko
LD [M] drivers/powercap/intel_rapl_msr.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/sysimgblt.ko
LD [M] drivers/video/fbdev/core/sysfillrect.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/psnap.ko
LD [M] net/802/stp.ko
LD [M] net/ipv6/ipv6.ko
LD [M] net/llc/llc.ko
LD [M] net/8021q/8021q.ko
LD [M] net/rfkill/rfkill.ko
LD [M] net/wireless/cfg80211.ko
LD [M] sound/core/snd-pcm.ko
LD [M] sound/core/snd-timer.ko
LD [M] sound/ac97_bus.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

real    0m54,475s
user    4m47,398s
sys     0m25,730s
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 2.7: Zakończenie pierwszego budowania modułów.

```

root@slack:/usr/src/linux-5.18.5# make -j6 modules_install
INSTALL /lib/modules/5.18.5-smp/kernel/arch/x86/crypto/crc32-pclmul.ko
INSTALL /lib/modules/5.18.5-smp/kernel/arch/x86/events/intel/intel-cstate.ko
INSTALL /lib/modules/5.18.5-smp/kernel/arch/x86/events/rapl.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/acpi/ac.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/acpi/button.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/acpi/video.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/block/loop.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/char/agp/intel-agp.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/char/agp/agpgart.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/char/agp/intel-gtt.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/drm.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/drm_kms_helper.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/drm_ttm_helper.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/ttm/ttm.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/vmwgfx/vmwgfx.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/i2c/algos/i2c-algo-bit.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/i2c/busses/i2c-piix4.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/i2c/i2c-core.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/input/evdev.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/input/joydev.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/input/mouse/psmouse.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/net/ethernet/amd/pcnet32.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/net/serio/serio_raw.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/net/mii.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/powercap/intel_rapl_common.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/powercap/intel_rapl_msr.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/video/fbdev/core/fb_sys_fops.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/video/fbdev/core/syscopyarea.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/video/fbdev/core/sysfillrect.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/video/fbdev/core/sysimgblt.ko

```

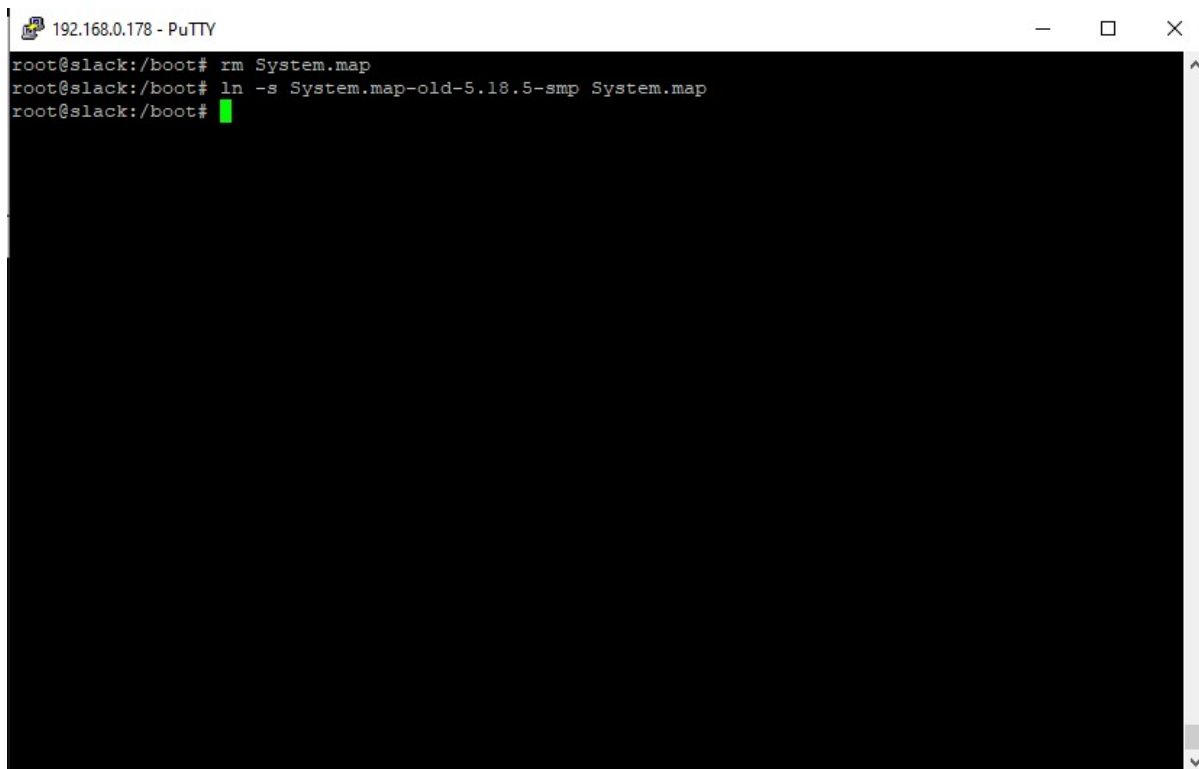
Rysunek 2.8: Pierwsza instalacja modułów.

Następnie do katalogu /boot przekopiowałem potrzebne pliki. Wszystkie komendy są widoczne na Rysunku 2.9.

Po przekopiowaniu przeszedłem do katalogu /boot i utworzyłem wiazanie symboliczne potrzebne do utworzenia dysku RAM(Rys. 2.10).

```
root@slack:/usr/src/linux-5.18.5# cp arch/x86/boot/bzImage /boot/vmlinuz-old-5.18.5-smp
root@slack:/usr/src/linux-5.18.5# cp System.map /boot/System.map-old-5.18.5-smp
root@slack:/usr/src/linux-5.18.5# cp .config /boot/config-old-5.18.5-smp
root@slack:/usr/src/linux-5.18.5#
```

Rysunek 2.9: Kopiowanie plików do /boot.



Rysunek 2.10: Tworzenie wiązania symbolicznego.

Następnie użyłem skryptu `mkinitrd_command_generator.sh`, podając do argumentu `-k` wersję jądra, w celu wygenerowania polecenia do utworzenia dysku RAM. Na Rysunku 2.11 widać otrzymanie i użycie tej komendy. Następnie dodaję nowe jądro do pliku konfiguracyjnego `lilo`. Nazywam go **Old-Kernel**(Rys. 2.12) i używam polecenia `lilo` w celu dodania nowej konfiguracji(Rys. 2.13).

Po poprawnym dodaniu nowej konfiguracji, uruchamiam maszynę ponownie. Po jej włączeniu wita mnie panel `lilo`. Jedną z opcji do wyboru jest dodany przeze mnie "Old-kernel"(Rys. 2.14). Po jego wybraniu, wita mnie on wiadomością "Welcome to Linux 5.18.5smp" czyli wersją którą pobierałem i kompilowałem.

```

192.168.0.178 - PuTTY
root@slack:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.5-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 5.18.5-smp -f ext4 -r /dev/sdal -m ext4 -u -o /boot/initrd.gz
root@slack:/boot# mkinitrd -c -k 5.18.5-smp -f ext4 -r /dev/sdal -m ext4 -u -o /boot/initrd-old-5.18.5-smp.gz
49038 bloków
/boot/initrd-old-5.18.5-smp.gz created.
Be sure to run lilo again if you use it.
root@slack:/boot#

```

Rysunek 2.11: Tworzenie dysku RAM

```

192.168.0.178 - PuTTY
GNU nano 6.0 /etc/lilo.conf
#vga=773
# VESA framebuffer console @ 800x600x64k
#vga=788
# VESA framebuffer console @ 800x600x32k
#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
root = /dev/sdal
label = "Slackware 15.0"
read-only

image = /boot/vmlinuz-old-5.18.5-smp
root = /dev/sdal
initrd = /boot/initrd-old-5.18.5-smp.gz
label = "Old-kernel"
read-only
# Linux bootable partition config ends

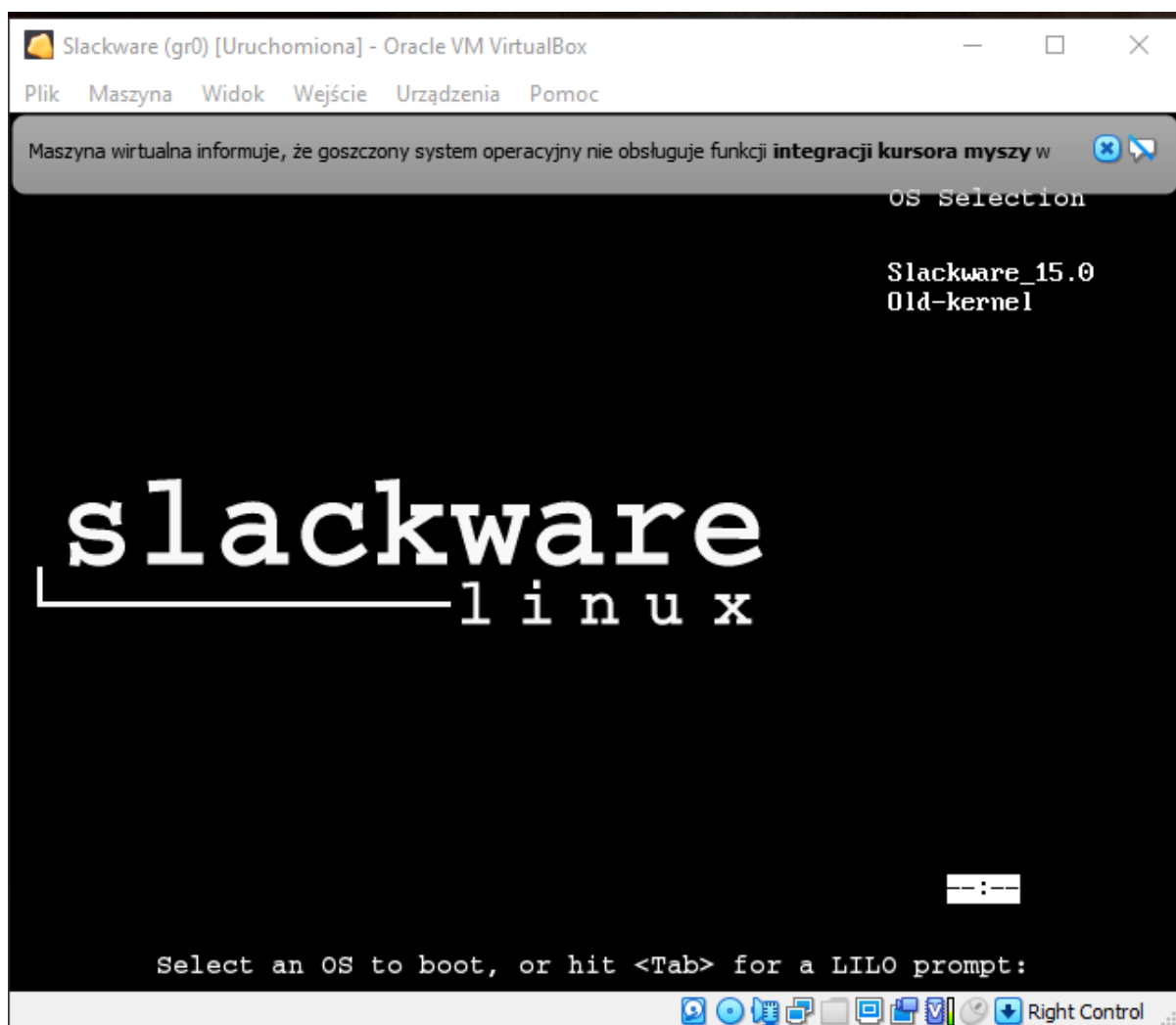
[ Zapisano 75 linii ]
^G Pomoc      ^O Zapisz      ^W Wyszukaj    ^K Wytnij     ^T Wykonaj    ^C Lokalizacja
^X Wyjdź      ^R Wczyt.plik ^\ Zastąp     ^U Wklej      ^J Wyjustuj   ^_ Do linii

```

Rysunek 2.12: Dodanie "Old-kernel" do pliku lilo.conf

```
root@slack:/boot# lilo
Warning: LBA32 addressing assumed
Added Slackware_15.0 *
Added Old-kernel +
One warning was issued.
root@slack:/boot# █
```

Rysunek 2.13: Użycie komendy lilo dla starej wersji.



Rysunek 2.14: Panel lilo po dodaniu old-kernel

```

Slackware (gr0) [Uruchomiona] - Oracle VM VirtualBox
Plik Maszyna Widok Wejście Urządzenia Pomoc
eth0: soliciting a DHCP lease
Maszyna wirtualna informuje, że goszczony system operacyjny obsługuje funkcję integracji kursora myszy. Oznacza to, że nie trzeba ręcznie przechwytywać kursora myszy aby móc go używać w
eth0: probing address 10.0.2.15/24
eth0: leased 10.0.2.15 for 86400 seconds
eth0: adding route to 10.0.2.0/24
eth0: adding default route via 10.0.2.2
forked to background, child pid 681
eth1: polling for DHCP server
dhcpcd-9.4.1 starting
DUID 00:04:d7:57:74:a9:cc:c2:48:91:ad:2d:c2:58:7e:16:3d:85
eth1: waiting for carrier
eth1: carrier acquired
eth1: IFAID 27:3d:3e:c4
eth1: soliciting a DHCP lease
eth1: offered 192.168.0.178 from 192.168.0.1
eth1: probing address 192.168.0.178/24
eth1: leased 192.168.0.178 for 86400 seconds
eth1: adding route to 192.168.0.0/24
eth1: adding default route via 192.168.0.1
forked to background, child pid 778
Starting system message bus: /usr/bin/dbus-uuidgen --ensure ; /usr/bin/dbus-daemon --system
Starting elogind: /lib/elogind/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 -l 1
Loading /usr/share/kbd/keymaps/i386/german/qwertz.pl.map.gz
Starting gpm: /usr/sbin/gpm -m /dev/mouse -t imps2

Welcome to Linux 5.18.5-smp i686 (tty1)

slack login: root
Password:
Last login: Tue Jun 21 18:54:46 on tty1
Linux 5.18.5-smp.
root@slack:~# ls -la
./          .bash_history  .dbus/        .hplip/       .local/       .wget-hsts    Dokumenty/   Pobrane/      Szablony/
./          .cache/       .gnupg/      .kde/         .serverauth.1201 .xinitrc     Muzyka/     Publiczny/    Video/
.Xauthority .config/     .gtkrc-2.0   .lessht      .serverauth.1540 .xsession*   Obrazy/     Pulpit/      hej_Kasia.txt
root@slack:~# $

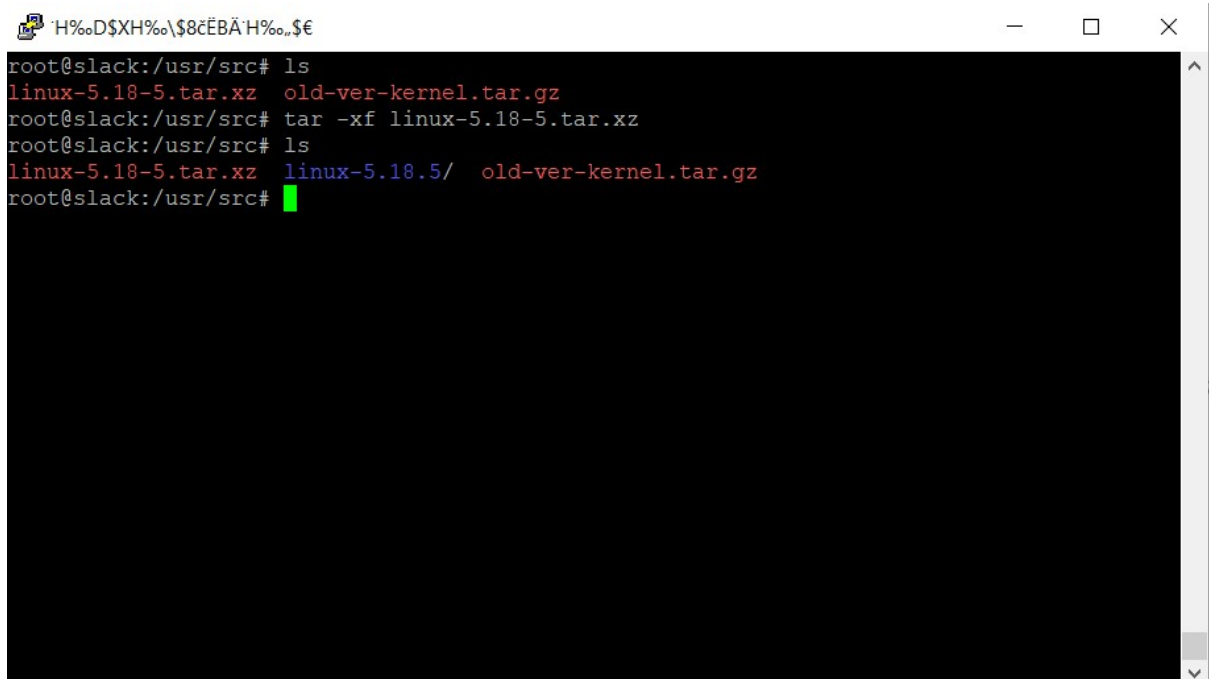
```

Rysunek 2.15: Old-kernel po uruchomieniu

Rozdział 3

Kompilacja jądra z wykorzystaniem nowej metody `streamline_config.pl`

Na początku ponownie wypakowuję pobrane jądro(Rys 3.1) Następnie kopiuję obecną



```
root@slack:/usr/src# ls
linux-5.18-5.tar.xz  old-ver-kernel.tar.gz
root@slack:/usr/src# tar -xf linux-5.18-5.tar.xz
root@slack:/usr/src# ls
linux-5.18-5.tar.xz  linux-5.18.5/  old-ver-kernel.tar.gz
root@slack:/usr/src#
```

Rysunek 3.1: Ponowne wypakowanie jądra

konfigurację oraz używam polecenia `streamline_config.pl`(Rys. 3.2). Zaraz po tym używam komendy `make oldconfig`(Rys. 3.3) i akceptuję wszystko jako domyślne(Rys.3.4).


```

root@slack:/usr/src/linux-5.18.5# cp /boot/config .config
root@slack:/usr/src/linux-5.18.5# ls -a
./          .gitignore  LICENSES/   crypto/     lib/        tools/
../         .mailmap    MAINTAINERS drivers/     mm/         usr/
.clang-format  COPYING     Makefile    fs/         net/        virt/
.cocciconfig   CREDITS     README      include/    samples/
.config        Documentation/ arch/        init/       scripts/
.get_maintainer.ignore Kbuild      block/      ipc/        security/
.gitattributes Kconfig     certs/      kernel/     sound/
root@slack:/usr/src/linux-5.18.5# scripts/kconfig/streamline_config.pl > config_strip
using config: '.config'
root@slack:/usr/src/linux-5.18.5# mv .config .config_backup
root@slack:/usr/src/linux-5.18.5# mv config_strip .config
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 3.2: Utworzenie nowej konfiguracji za pomocą streamline_config.pl

```

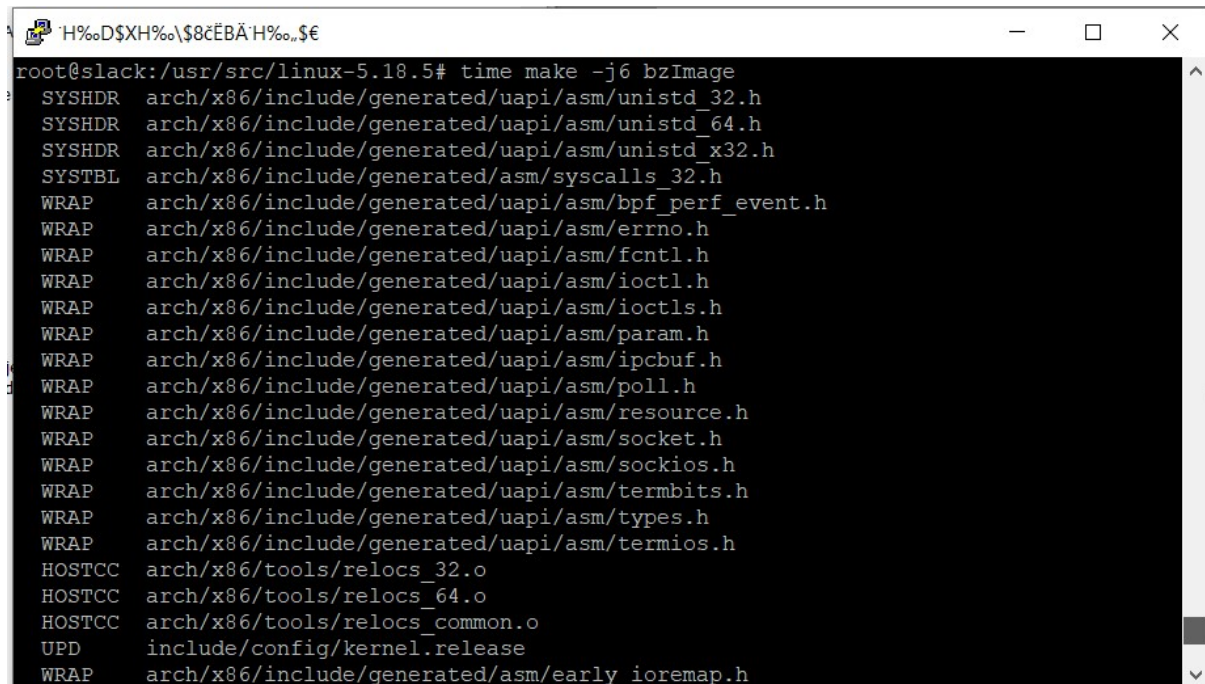
Test module for compilation of bitops operations (TEST_BITOPS) [N/m/?] n
Test module for stress/performance analysis of vmalloc allocator (TEST_VMALLOC) [N/m/?]
n
Test user/kernel boundary protections (TEST_USER_COPY) [N/m/?] n
Test BPF filter functionality (TEST_BPF) [N/m/?] n
Test blackhole netdev 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-5.18.5#
root@slack:/usr/src/linux-5.18.5#
root@slack:/usr/src/linux-5.18.5#
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 3.3: Zostawienie wszystkich ustawień jako domyślnych

Po utworzeniu konfiguracji zaczynam kompilację jądra. Ponownie używam polecenia *time make -j6 bzImage*. Kompilacja trwa 9 minut i 23 sekundy, czyli niewiele więcej niż przy starej metodzie (Rys. 3.5). Następnie zaczynam budować moduły (Rys. 3.6) i trwa

to 58 sekund, czyli prawie tyle samo co w przypadku starej metody(Rys. 3.7). Po ich zbudowaniu, użyłem polecenia *make modules_install* do ich zainstalowania(Rys. 3.8).

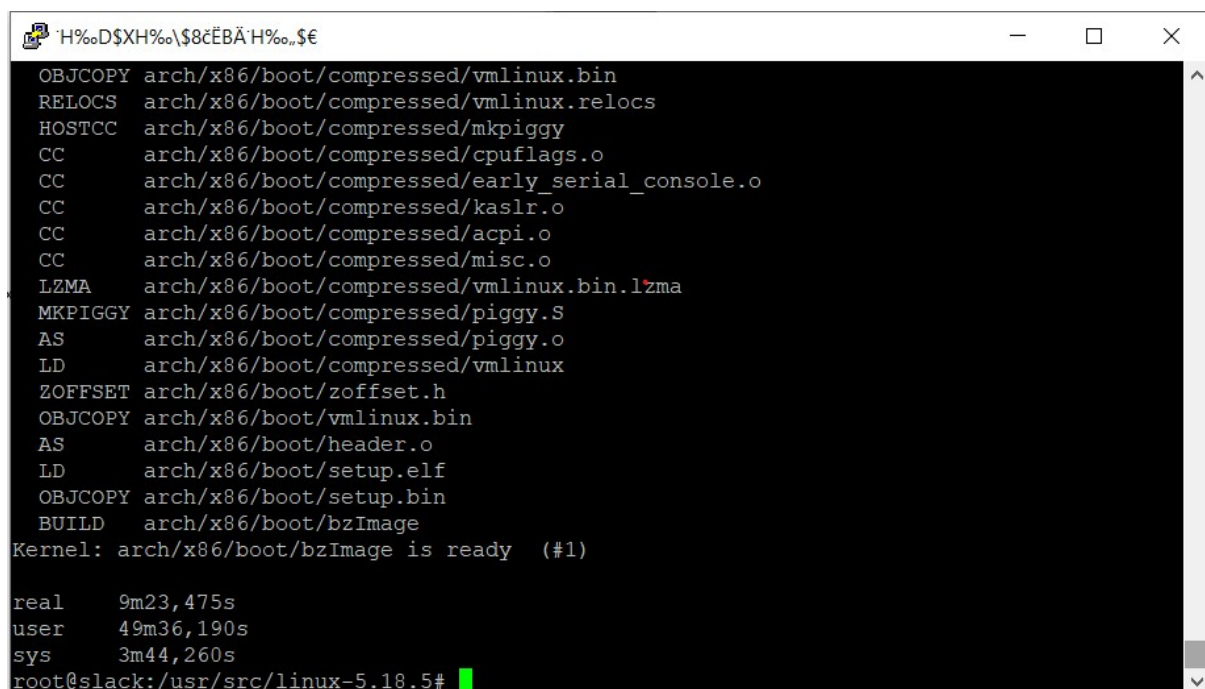


```

root@slack:/usr/src/linux-5.18.5# time 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
WRAP arch/x86/include/generated/uapi/asm/param.h
WRAP arch/x86/include/generated/uapi/asm/ipcbuf.h
WRAP arch/x86/include/generated/uapi/asm/poll.h
WRAP arch/x86/include/generated/uapi/asm/resource.h
WRAP arch/x86/include/generated/uapi/asm/socket.h
WRAP arch/x86/include/generated/uapi/asm/sockios.h
WRAP arch/x86/include/generated/uapi/asm/termbits.h
WRAP arch/x86/include/generated/uapi/asm/types.h
WRAP arch/x86/include/generated/uapi/asm/termios.h
HOSTCC arch/x86/tools/relocs_32.o
HOSTCC arch/x86/tools/relocs_64.o
HOSTCC arch/x86/tools/relocs_common.o
UPD include/config/kernel.release
WRAP arch/x86/include/generated/asm/early_ioremap.h

```

Rysunek 3.4: Rozpoczęcie kompilacji jądra z nową konfiguracją.



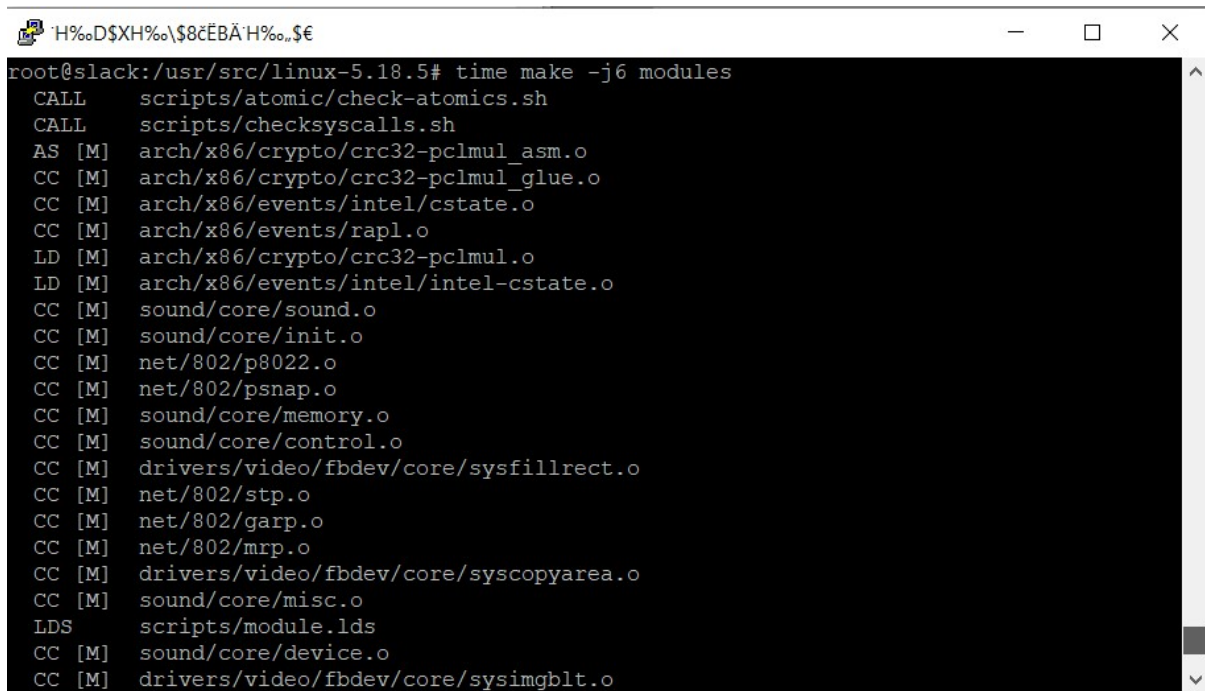
```

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)

real    9m23,475s
user    49m36,190s
sys     3m44,260s
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 3.5: Zakończenie kompilacji jądra z nową konfiguracją.

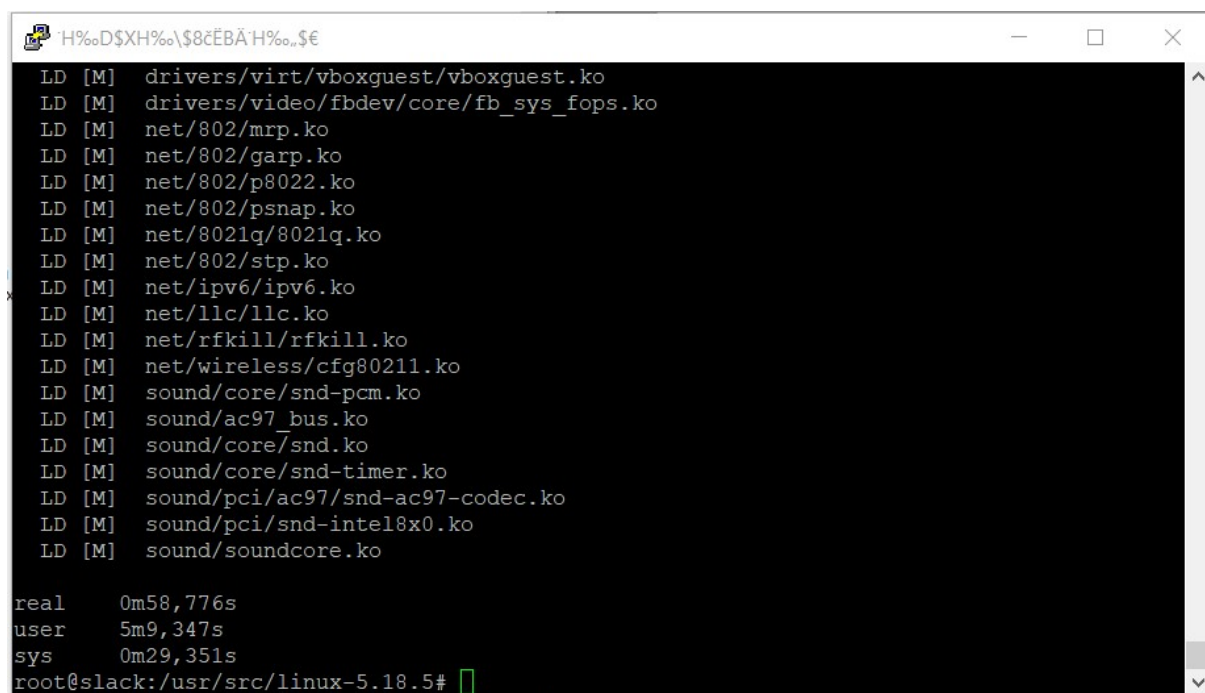


```

root@slack:/usr/src/linux-5.18.5# time make -j6 modules
CALL      scripts/atomic/check-atomics.sh
CALL      scripts/checksyscalls.sh
AS [M]    arch/x86/crypto/crc32-pclmul_asm.o
CC [M]    arch/x86/crypto/crc32-pclmul_glue.o
CC [M]    arch/x86/events/intel/cstate.o
CC [M]    arch/x86/events/rapl.o
LD [M]    arch/x86/crypto/crc32-pclmul.o
LD [M]    arch/x86/events/intel/intel-cstate.o
CC [M]    sound/core/sound.o
CC [M]    sound/core/init.o
CC [M]    net/802/p8022.o
CC [M]    net/802/psnap.o
CC [M]    sound/core/memory.o
CC [M]    sound/core/control.o
CC [M]    drivers/video/fbdev/core/sysfillrect.o
CC [M]    net/802/stp.o
CC [M]    net/802/garp.o
CC [M]    net/802/mrp.o
CC [M]    drivers/video/fbdev/core/syscopyarea.o
CC [M]    sound/core/misc.o
LDS       scripts/module.lds
CC [M]    sound/core/device.o
CC [M]    drivers/video/fbdev/core/sysimgblt.o

```

Rysunek 3.6: Budowanie modułów z nową metodą



```

LD [M]    drivers/virt/vboxguest/vboxguest.ko
LD [M]    drivers/video/fbdev/core/fb_sys_fops.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/8021q/8021q.ko
LD [M]    net/802/stp.ko
LD [M]    net/ipv6/ipv6.ko
LD [M]    net/llc/llc.ko
LD [M]    net/rfkill/rfkill.ko
LD [M]    net/wireless/cfg80211.ko
LD [M]    sound/core/snd-pcm.ko
LD [M]    sound/ac97_bus.ko
LD [M]    sound/core/snd.ko
LD [M]    sound/core/snd-timer.ko
LD [M]    sound/pci/ac97/snd-ac97-codec.ko
LD [M]    sound/pci/snd-intel8x0.ko
LD [M]    sound/soundcore.ko

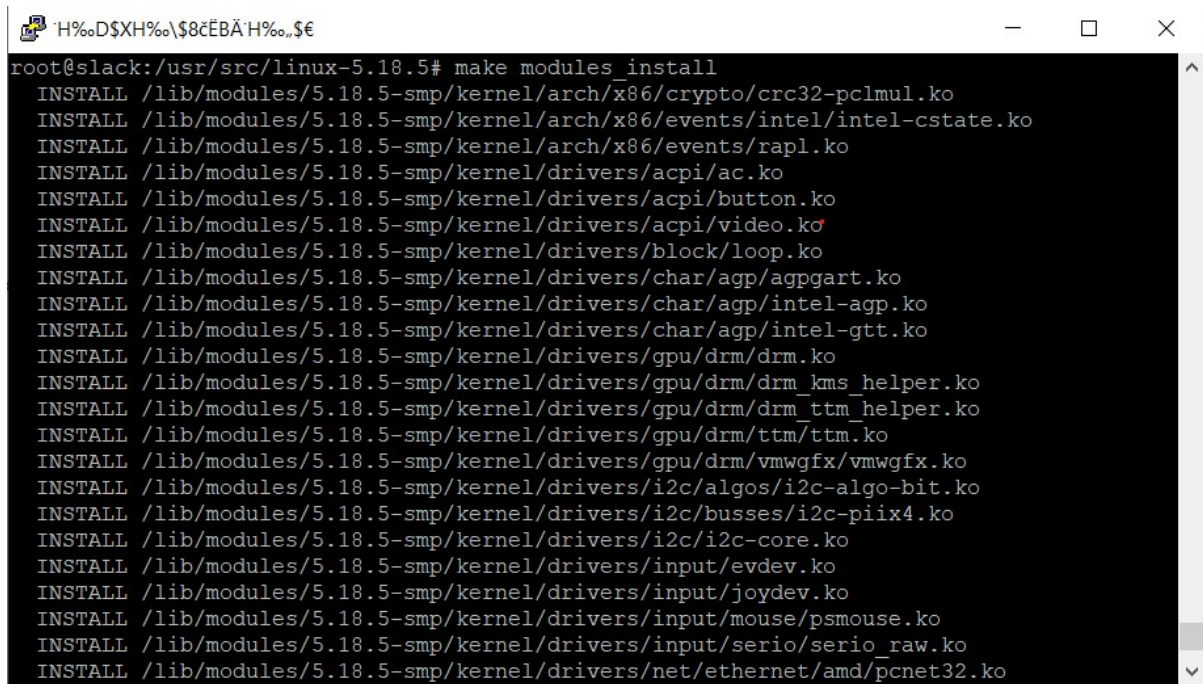
real      0m58,776s
user      5m9,347s
sys       0m29,351s
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 3.7: Zakończenie budowania modułów z nową metodą

Po tym wszystkim, przechodzę do kopiowania potrzebnych plików tak jak w przypadku starej metody i tak samo tworzę dowiązanie(Rys. 3.9).

Następnie, tak samo jak poprzednio, tworzę dysk RAM(Rys. 3.10). Dodaję nowe jądro do pliku konfiguracyjnego lilo nazywając je "New-kernel" (Rys. 3.11) i włączam polecenie

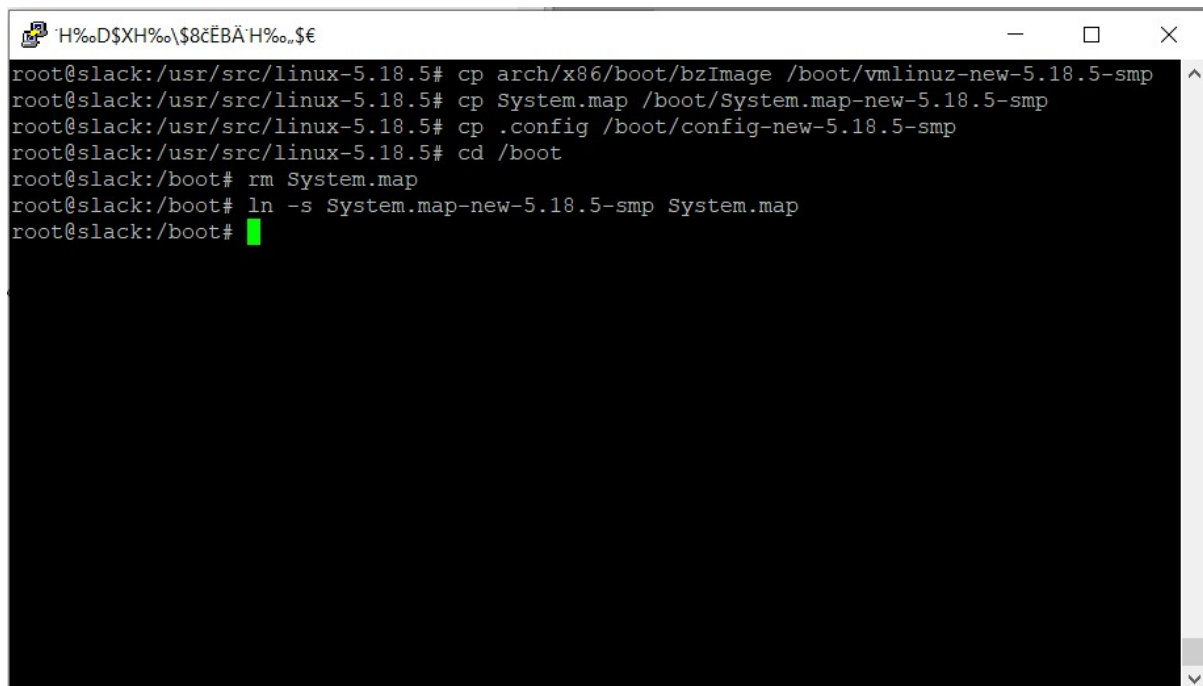


```

root@slack:/usr/src/linux-5.18.5# make modules install
INSTALL /lib/modules/5.18.5-smp/kernel/arch/x86/crypto/crc32-pclmul.ko
INSTALL /lib/modules/5.18.5-smp/kernel/arch/x86/events/intel/intel-cstate.ko
INSTALL /lib/modules/5.18.5-smp/kernel/arch/x86/events/rapl.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/acpi/ac.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/acpi/button.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/acpi/video.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/block/loop.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/char/agp/agpgart.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/char/agp/intel-agp.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/char/agp/intel-gtt.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/drm.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/drm_kms_helper.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/drm_ttm_helper.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/ttm/ttm.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/gpu/drm/vmwgfx/vmwgfx.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/i2c/algos/i2c-algo-bit.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/i2c/busses/i2c-piix4.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/i2c/i2c-core.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/input/evdev.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/input/joydev.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/input/mouse/psmouse.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/input/serio/serio_raw.ko
INSTALL /lib/modules/5.18.5-smp/kernel/drivers/net/ethernet/amd/pcnet32.ko

```

Rysunek 3.8: Instalacja modułów z nową metodą



```

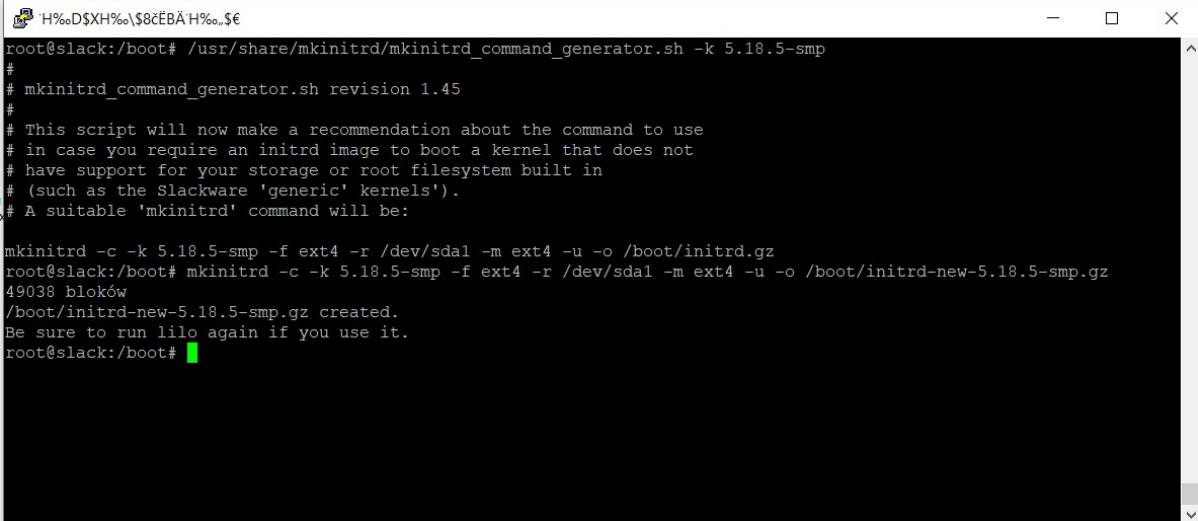
root@slack:/usr/src/linux-5.18.5# cp arch/x86/boot/bzImage /boot/vmlinuz-new-5.18.5-smp
root@slack:/usr/src/linux-5.18.5# cp System.map /boot/System.map-new-5.18.5-smp
root@slack:/usr/src/linux-5.18.5# cp .config /boot/config-new-5.18.5-smp
root@slack:/usr/src/linux-5.18.5# cd /boot
root@slack:/boot# rm System.map
root@slack:/boot# ln -s System.map-new-5.18.5-smp System.map
root@slack:/boot#

```

Rysunek 3.9: Kopiowanie potrzebnych plików i tworzenie dowiązania

lilo w celu dodania nowej konfiguracji (Rys.3.12).

Na sam koniec uruchamiam ponownie wirtualną maszynę. Po jej włączenia wita mnie panel lilo z 3 opcjami do wyboru w tym nowo dodany "New-kernel" (Rys. 3.13). Po jego wybraniu wszystko działa prawidłowo, linux wita mnie z poprawną wersją. W tym wy-

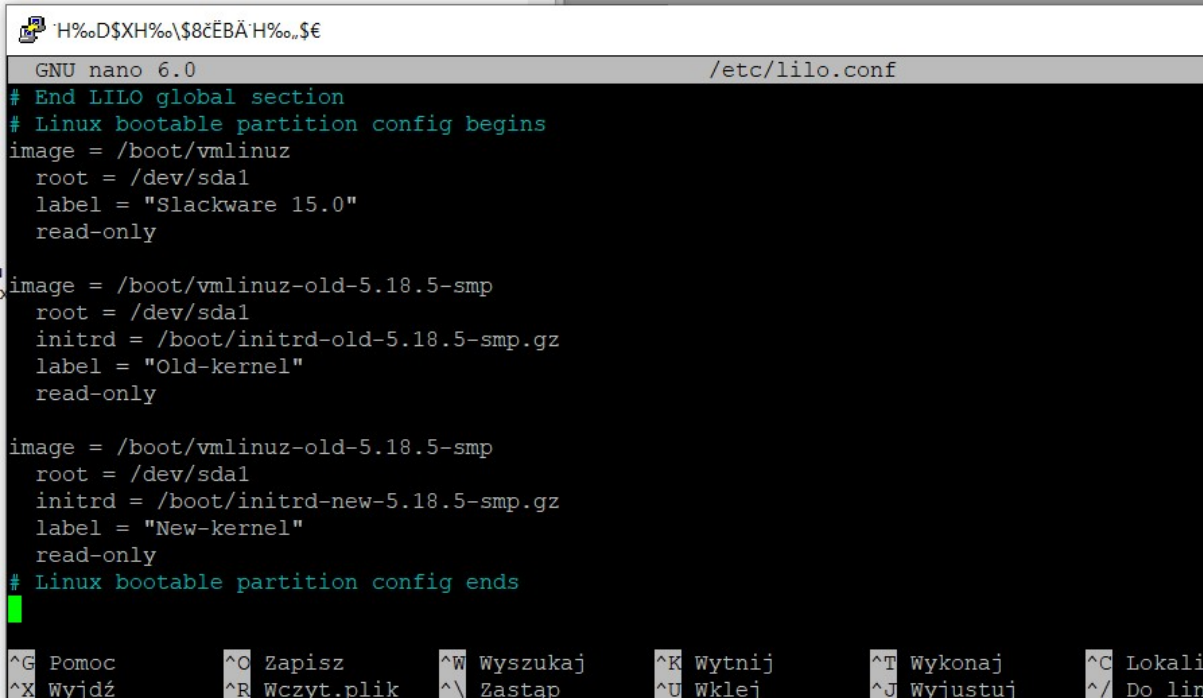


```

root@slack:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.5-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 5.18.5-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
root@slack:/boot# mkinitrd -c -k 5.18.5-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-new-5.18.5-smp.gz
49038 bloków
/boot/initrd-new-5.18.5-smp.gz created.
Be sure to run lilo again if you use it.
root@slack:/boot#

```

Rysunek 3.10: Tworzenie dysku RAM



```

GNU nano 6.0 /etc/lilo.conf
# End LILO global section
# Linux bootable partition config begins
image = /boot/vmlinuz
  root = /dev/sda1
  label = "Slackware 15.0"
  read-only

image = /boot/vmlinuz-old-5.18.5-smp
  root = /dev/sda1
  initrd = /boot/initrd-old-5.18.5-smp.gz
  label = "Old-kernel"
  read-only

image = /boot/vmlinuz-old-5.18.5-smp
  root = /dev/sda1
  initrd = /boot/initrd-new-5.18.5-smp.gz
  label = "New-kernel"
  read-only
# Linux bootable partition config ends

```

[^]G Pomoc [^]O Zapisz [^]W Wyszukaj [^]K Wytnij [^]T Wykonaj [^]C Lokalizuj
[^]X Wyjdź [^]R Wczyt.plik [^]\ Zastap [^]U Wklej [^]J Wyjustuj [^]/ Do linii

Rysunek 3.11: Dodanie "New-kernel" do pliku konfiguracyjnego lilo

padku dodatkowo użyłem *uname -a* dla upewnienia się, że wersja jądra jest taka jaką kompilowałem. (Rys. 3.14).

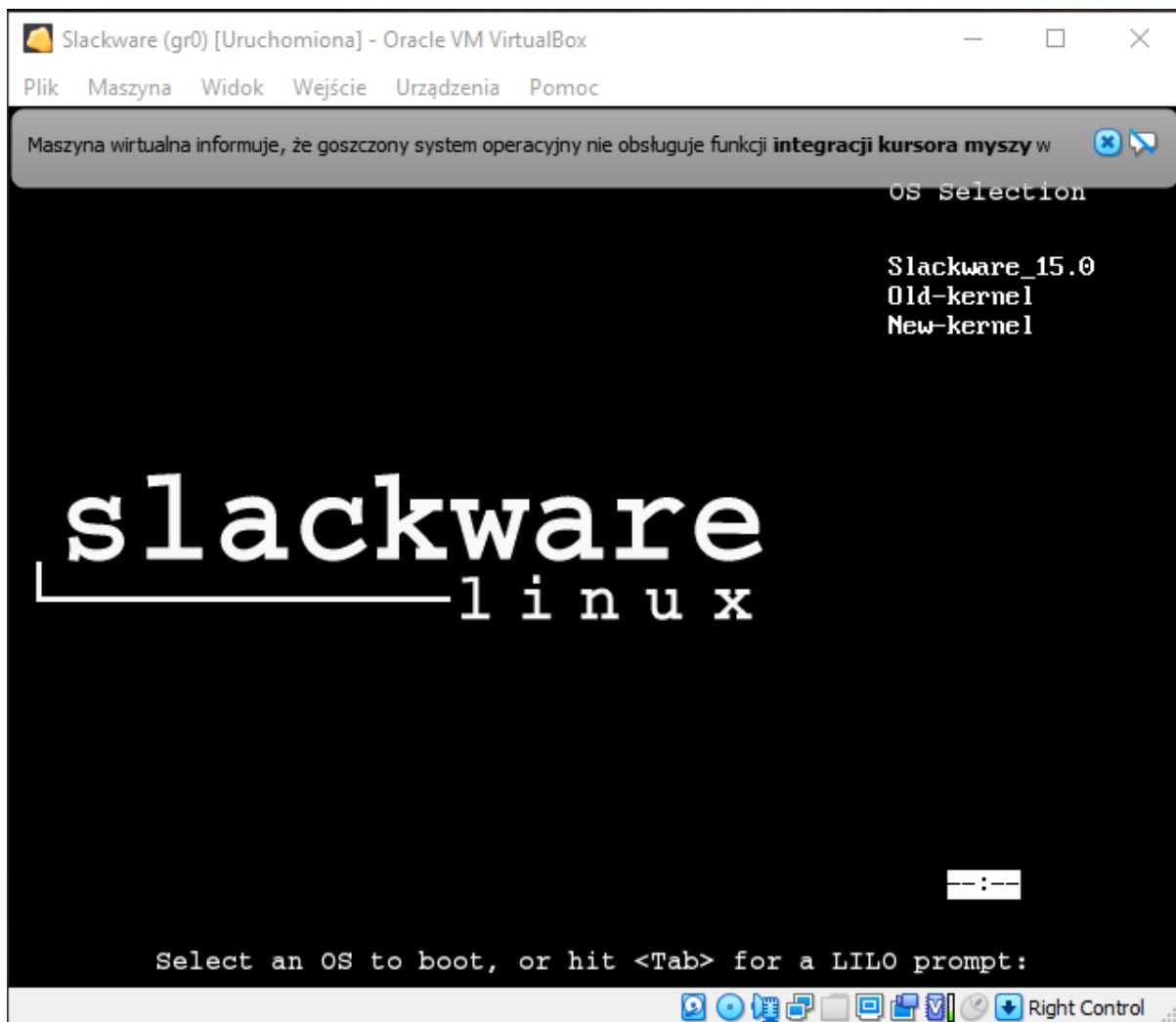


```

root@slack:/boot# lilo
Warning: LBA32 addressing assumed
Added Slackware_15.0 *
Added Old-kernel +
Added New-kernel +
One warning was issued.
root@slack:/boot# █

```

Rysunek 3.12: Dodanie nowej konfiguracji do lilo



Rysunek 3.13: Panel wyboru lilo z nowy "New-kernel"

```

Slackware (gr0) [Uruchomiona] - Oracle VM VirtualBox
Plik Maszyna Widok Wejście Urządzenia Pomoc
eth0: IFAID 27:61:7a:e9
Maszyna wirtualna informuje, że goszczony system operacyjny obsługuje funkcję integracji kursora myszy. Oznacza to, że nie trzeba ręcznie przechwytywać kursora myszy aby móc go używać w
eth0: offered 10.0.2.15 from 10.0.2.2
eth0: probing address 10.0.2.15/24
eth0: leased 10.0.2.15 for 86400 seconds
eth0: adding route to 10.0.2.0/24
eth0: adding default route via 10.0.2.2
forked to background, child pid 680
eth1: polling for DHCP server
dhcpd-9.4.1 starting
DUID 00:04:d7:57:74:a9:cc:c2:48:91:ad:2d:c2:58:7e:16:3d:85
eth1: waiting for carrier
eth1: carrier acquired
eth1: IFAID 27:3d:3e:c4
eth1: soliciting a DHCP lease
eth1: offered 192.168.0.178 from 192.168.0.1
eth1: probing address 192.168.0.178/24
eth1: leased 192.168.0.178 for 86400 seconds
eth1: adding route to 192.168.0.0/24
eth1: adding default route via 192.168.0.1
forked to background, child pid 773
Starting system message bus: /usr/bin/dbus-uuidgen --ensure ; /usr/bin/dbus-daemon --system
Starting elogind: /lib/elogind/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 -l 1
Loading /usr/share/kbd/keymaps/i386/qwerty/pl.map.gz
Starting gpm: /usr/sbin/gpm -m /dev/mouse -t imps2

Welcome to Linux 5.18.5-smp i686 (tty1)

slack login: root
Password:
Last login: Tue Jun 21 23:37:45 on tty1
Linux 5.18.5-smp.
root@slack:~# uname -a
Linux slack.localhost 5.18.5-smp #1 SMP PREEMPT_DYNAMIC Tue Jun 21 18:20:52 CEST 2022 i686 Intel(R) Core(TM) i7-8700 CPU @ 3.20G
Hz GenuineIntel GNU/Linux
root@slack:~#

```

Rysunek 3.14: "New-kernel" po wybraniu

Rozdział 4

Wnioski końcowe

Po przeprowadzeniu kompilacji z obiema metodami na utworzenie konfiguracji, nie zauważam wielu różnic. Kompilacja z użyciem starej metody wypadła szybciej, jednak różnica 40 sekund nie jest znacząca. Dodatkową różnicą było zaimportowanie .config z innych miejsc. Obie metody są dla mnie podobne i nie miałbym problemu z ponownym przeprowadzeniem obu z nich.

Spis rysunków

1.1	Pobranie najnowszej wersji kernela	1
1.2	Rozpakowanie archiwum	2
2.1	Skopiowanie aktualnej konfiguracji	3
2.2	wykonanie polecenia <code>make localmodconfig</code>	4
2.3	Akceptacja domyślnych ustawień	4
2.4	Pierwsza kompilacja jądra z komendą <code>time</code>	5
2.5	Zakończenie pierwszej kompilacji	6
2.6	Pierwsze budowanie modułów.	6
2.7	Zakończenie pierwszego budowania modułów.	7
2.8	Pierwsza instalacja modułów.	7
2.9	Kopiowanie plików do <code>/boot</code>	8
2.10	Tworzenie wiązania symbolicznego.	8
2.11	Tworzenie dysku RAM	9
2.12	Dodanie "Old-kernel" do pliku <code>lilo.conf</code>	9
2.13	Użycie komendy <code>lilo</code> dla starej wersji.	10
2.14	Panel <code>lilo</code> po dodaniu <code>old-kernel</code>	10
2.15	Old-kernel po uruchomieniu	11
3.1	Ponowne wypakowanie jądra	12
3.2	Utworzenie nowej konfiguracji za pomocą <code>streamline.config.pl</code>	13
3.3	Zostawienie wszystkich ustawień jako domyślnych	13
3.4	Rozpoczęcie kompilacji jądra z nową konfiguracją.	14
3.5	Zakończenie kompilacji jądra z nową konfiguracją.	14
3.6	Budowanie modułów z nową metodą	15
3.7	Zakończenie budowania modułów z nową metodą	15
3.8	Instalacja modułów z nową metodą	16
3.9	Kopiowanie potrzebnych plików i tworzenie dowiązania	16
3.10	Tworzenie dysku RAM	17
3.11	Dodanie "New-kernel" do pliku konfiguracyjnego <code>lilo</code>	17
3.12	Dodanie nowej konfiguracji do <code>lilo</code>	18

3.13 Panel wyboru lilo z nowy "New-kernel"	18
3.14 "New-kernel" po wybraniu	19