

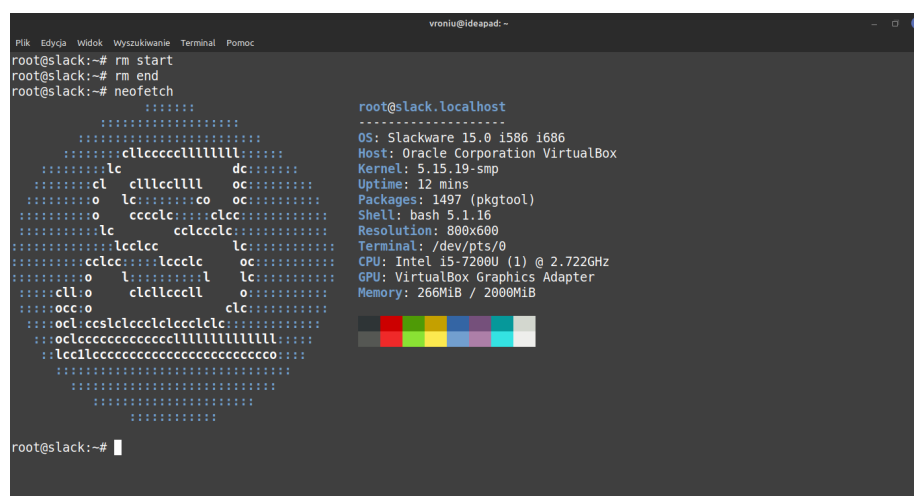
Kompilacja jądra Linux

Grzegorz Wrona

6 czerwca 2022

1 Przygotowanie

Przed aktualizacją na mojej wirtualnej maszynie był zainstalowany Slackware 15.0 z jądrem w wersji **5.15.19**.



```
root@slack:~# rm start
root@slack:~# rm end
root@slack:~# neofetch

root@slack.localhost
OS: Slackware 15.0 1586 i686
Host: Oracle Corporation VirtualBox
Kernel: 5.15.19-smp
Uptime: 12 mins
Packages: 1497 (pkgtool)
Shell: bash 5.1.16
Resolution: 800x600
Terminal: /dev/pts/0
CPU: Intel i5-7200U (1) @ 2.72GHz
GPU: VirtualBox Graphics Adapter
Memory: 266MiB / 2000MiB
```

Rysunek 1: Jak widać po wyniku komendy **neofetch**, wersja kernela to **5.15.19-smp**

W momencie pisania tego reportu najnowszą stabilną wersją kernela to **5.18.2**. Informacje i linki do pobrania najnowszych wersji jądra można znaleźć na stronie kernel.org. Stamtąd też skopiowałem link do wersji 5.18.2 i pobrałem go na moją wirtualną maszynę, korzystając z polecenia **wget**:

```
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
root@slack:~# wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.2.tar.xz
--2022-06-06 18:07:39-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.2.tar.xz
Resolving cdn.kernel.org (cdn.kernel.org)... 151.101.13.176, 2a04:4e42:1b::432
Connecting to cdn.kernel.org (cdn.kernel.org)[151.101.13.176]:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 129796020 (124M) [application/x-xz]
Saving to: 'linux-5.18.2.tar.xz'

linux-5.18.2.tar.xz      100%[=====] 123.78M  3.05MB/s   in 30s
2022-06-06 18:08:09 (4.13 MB/s) - 'linux-5.18.2.tar.xz' saved [129796020/129796020]

root@slack:~#
```

Rysunek 2: Wynik komendy `wget`

Pobrane archiwum przeniosłem do katalogu `/usr/src` i wypakowałem.

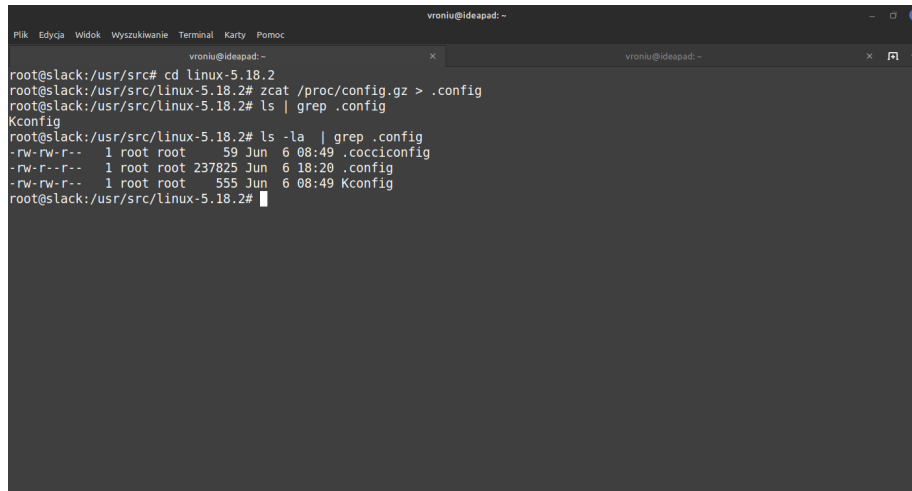
```
Plik Edycja Widok Wyszukiwanie Terminal Pomoc
root@slack:~# ls
linux-5.18.2.tar.xz
root@slack:~# mv linux-5.18.2.tar.xz /usr/src/
root@slack:~# ls
root@slack:~# cd /usr/src/
root@slack:/usr/src# tar -xpf linux-5.18.2.tar.xz
root@slack:/usr/src# ls
linux-5.18.2/  linux-5.18.2.tar.xz
root@slack:/usr/src#
```

Rysunek 3: Przeniesienie i wypakowanie archiwum. Nie korzystałem z przełącznika `-v`, aby nie widzieć na standardowym wyjściu nazw wypakowywanych plików.

Po wypakowaniu archiwum możemy przejść do kompilacji jądra.

2 Kompilacja - metoda stara

Przechodzimy do folderu z rozpakowanymi plikami korzystając z polecenia `cd` i rozpoczynamy konfigurację od skopiowania konfiguracji ze starego jądra do pliku `.config`. Wykorzystamy do tego komendę `zcat`:



```
vraniu@ideapad: ~  
Plik  Edycja  Widok  Wyszukiwanie  Terminal  Karty  Pomoc  
vraniu@ideapad: ~  
root@slack:/usr/src# cd linux-5.18.2  
root@slack:/usr/src/linux-5.18.2# zcat /proc/config.gz > .config  
root@slack:/usr/src/linux-5.18.2# ls | grep .config  
Kconfig  
root@slack:/usr/src/linux-5.18.2# ls -la | grep .config  
-rw-rw-r-- 1 root root   59 Jun  6 08:49 .coociconfig  
-rw-rw-r-- 1 root root 237825 Jun  6 18:20 .config  
-rw-rw-r-- 1 root root   555 Jun  6 08:49 Kconfig  
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 4: Skopiowanie starej konfiguracji - sprawdziłem czy się skopiowała z pomocą grepa

Następnie wywołujemy komendę `make localmodconfig`, która przygotowuje plik konfiguracyjny. Po jakimś czasie na ekranie zobaczymy komunikaty pytające o zaawansowane ustawienia - zostawiam na domyślne.

```
vrniu@ideapad: ~  
5. Pentium-MMX (M586MMX)  
6. Pentium-Pro (M686)  
7. Pentium-II/Celeron(pre-Coppermine) (MPENTIUMII)  
8. Pentium-III/Celeron(Coppermine)/Pentium-III Xeon (MPENTIUMIII)  
> 9. Pentium M (MPENTIUMM)  
10. Pentium-4/Celeron(P4-based)/Pentium-4 M/older Xeon (MPENTIUM4)  
11. K6/K6-II/K6-III (MK6)  
12. Athlon/Duron/K7 (MK7)  
13. Opteron/Athlon64/Hammer/K8 (MK8)  
14. Crusoe (MCRUSOE)  
15. Efficeon (MEFFICEON)  
16. Winchip-C6 (MWINCHIPC6)  
17. Winchip-2/Winchip-2A/Winchip-3 (MWINCHIP3D)  
18. AMD Elan (MELAN)  
19. GeodeGX1 (MGEODEGX1)  
20. Geode GX/LX (MGEODE LX)  
21. CyrixIII/VIA-C3 (MCYRIXIII)  
22. VIA C3-2 (Nehemiah) (MVIAC3_2)  
23. VIA C7 (MVIAC7)  
24. Core 2/newer Xeon (MCORE2)  
25. Intel Atom (MATOM)  
choice[1-25]: 8  
Generic x86 support (X86_GENERIC) [Y/n/?] y  
HPET Timer Support (HPET_TIMER) [Y/n/?] y  
Enable DMI scanning (DMI) [Y/n/?] y  
Maximum number of CPUs (NR_CPUS) [32] 32  
Cluster scheduler support (SCHED_CLUSTER) [Y/n/?] (NEW) █
```

Rysunek 5: Nie wiem o co chodzi to klikam enter

Po przeklikaniu opcji dostajemy wynik komendy i informację, że konfiguracja została zapisana do pliku **.config**

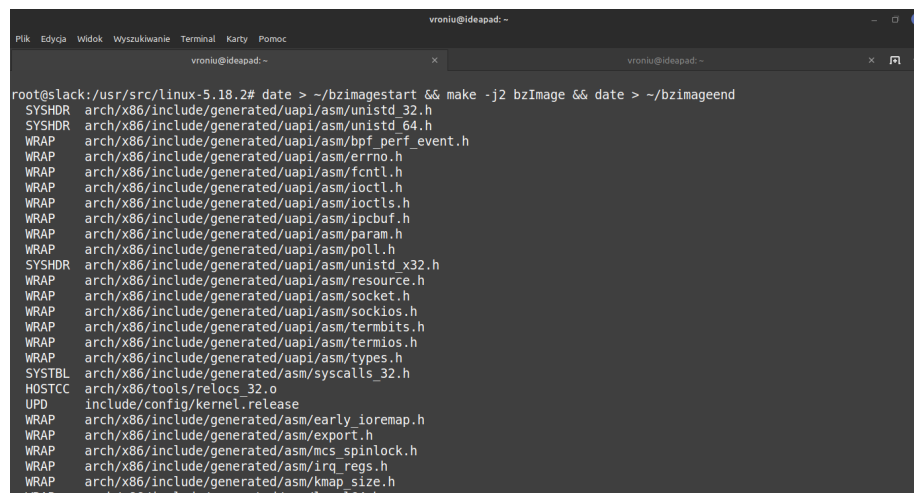
```
vrniu@ideapad: ~  
Test bitmap *() family of functions at runtime (TEST_BITMAP) [N/m/y/?] n  
Test functions located in the uuid module at runtime (TEST_UUID) [N/m/y/?] n  
Test the XArray code at runtime (TEST_XARRAY) [N/m/y/?] n  
Perform selftest on resizable hash table (TEST_RHASHTABLE) [N/m/y/?] 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_VMALLO) [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-5.18.2# █
```

Rysunek 6: Końcowy wynik komendy **make localmodconfig**

Pora na proces kompilacji obrazu jądra. Z racji że moja maszyna wirtualna posiada przydzielony tylko jeden rdzeń (co widać na zrzucie ekranu 1 w parametrze **CPU**), uruchomię komendę **make bzImage** z parametrem **-j2**, co powinno nieco przyspieszyć proces kompilacji. Postanowiłem też zmierzyć, ile zajmie proces kompilacji - w tym celu wymyśliłem takie polecenie:

```
date > ~/bzimagestart && make -j2 bzImage && date > ~/bzimageend
```

Przekierowanie komendy **date** powinno zapisać do plików w katalogu domowym datę bezpośrednio przed rozpoczęciem i bezpośrednio po zakończeniu kompilacji. Uruchamiam komendę i pozostaje tylko czekać.



```

vroniu@ideapad: ~
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc

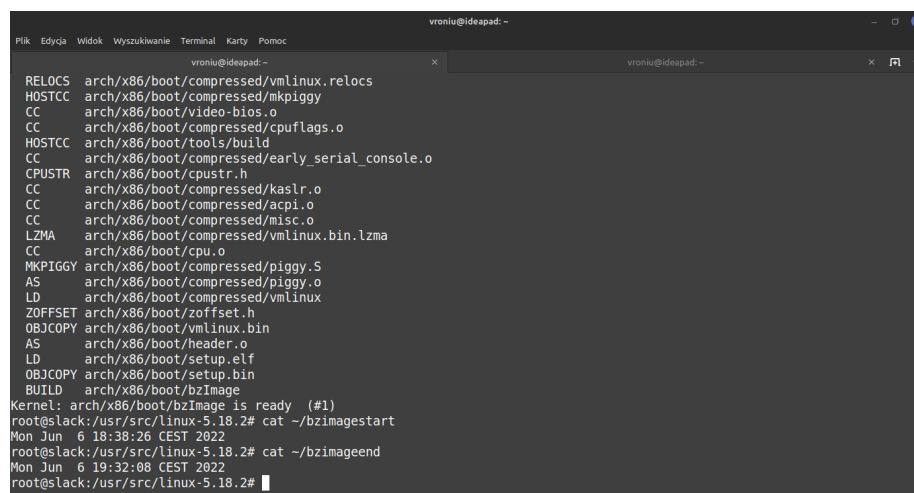
vroniu@ideapad: ~
vroniu@ideapad: ~

root@slack:/usr/src/linux-5.18.2# date > ~/bzimagestart && make -j2 bzImage && date > ~/bzimageend
SYSHDR arch/x86/include/generated/uapi/asm/unistd_32.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_64.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/poll.h
SYSHDR arch/x86/include/generated/uapi/asm/unistd_x32.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/termios.h
WRAP arch/x86/include/generated/uapi/asm/types.h
SYSTBL arch/x86/include/generated/asm/syscalls_32.h
HOSTCC arch/x86/tools/relocs_32.o
UPD include/config/kernel.release
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 7: Komenda, której użyłem oraz rozpoczęcie procesu kompilacji obrazu jądra

Po zakończeniu działania komendy dostajemy informację o ścieżce, gdzie znajduje się obraz jądra. Sprawdzmy, czy w plikach zapisały się daty rozpoczęcia i zakończenia kompilacji.



```

vroniu@ideapad: ~
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc

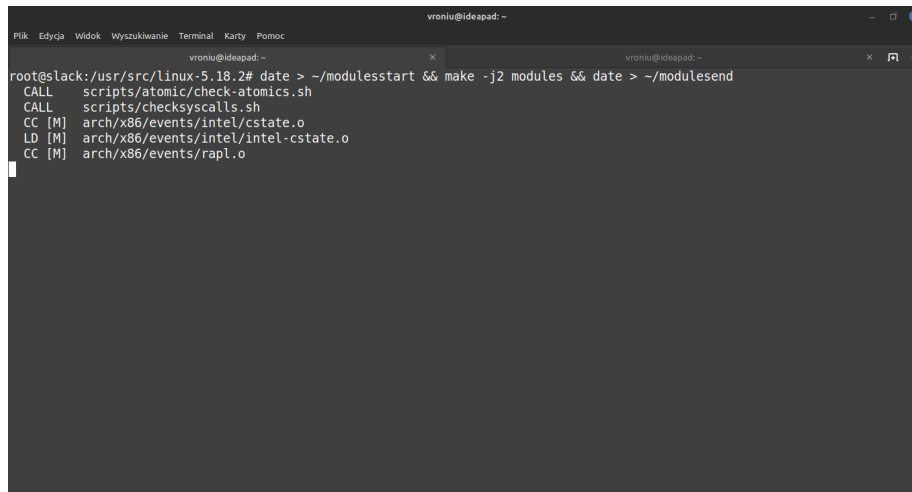
vroniu@ideapad: ~
vroniu@ideapad: ~

RELCO arch/x86/boot/compressed/vmlinux.relco
HOSTCC arch/x86/boot/compressed/mkpiggy
CC arch/x86/boot/video-bios.o
CC arch/x86/boot/compressed/cpuflags.o
HOSTCC arch/x86/boot/tools/build
CC arch/x86/boot/compressed/early_serial_console.o
CPUPSTR arch/x86/boot/cpustr.h
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
CC arch/x86/boot/cpu.o
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-5.18.2# cat ~/bzimagestart
Mon Jun 6 18:38:26 CEST 2022
root@slack:/usr/src/linux-5.18.2# cat ~/bzimageend
Mon Jun 6 19:32:08 CEST 2022
root@slack:/usr/src/linux-5.18.2#

```

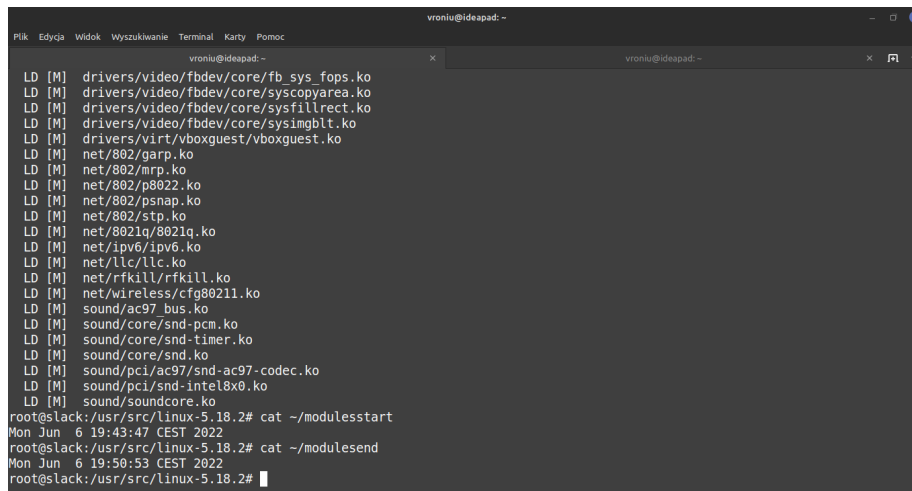
Rysunek 8: Kompilacja obrazu jądra zajęła około 54 minuty.

Teraz pora na zbudowanie modułów jądra z użyciem komendy `make modules` - również z zapisaniem czasu rozpoczęcia i zakończenia i parametrem `-j2`.



```
vraniu@ideapad: ~  
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc  
vraniu@ideapad: ~  
root@slack:/usr/src/linux-5.18.2# date > ~/modulesstart && make -j2 modules && date > ~/modulesend  
CALL scripts/atomic/check-atomics.sh  
CALL scripts/checksyscalls.sh  
CC [M] arch/x86/events/intel/cstate.o  
LD [M] arch/x86/events/intel/intel-cstate.o  
CC [M] arch/x86/events/rapl.o
```

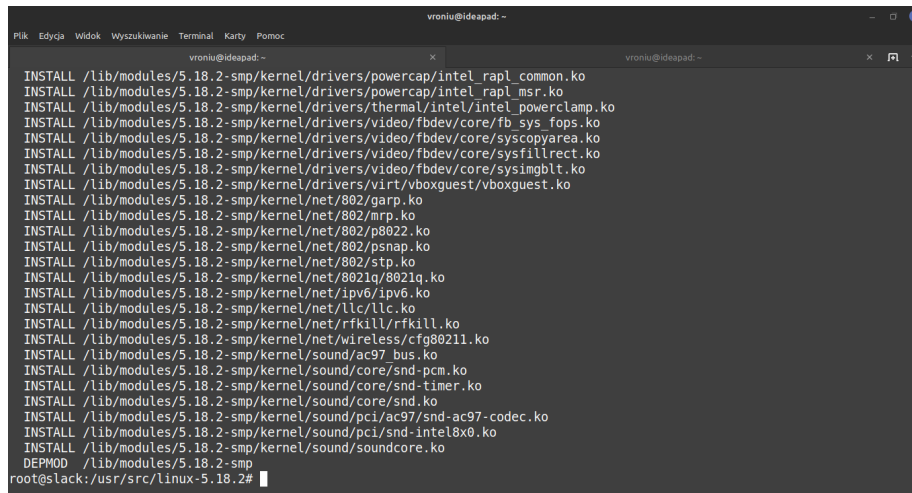
Rysunek 9: Wywołanie komendy do zbudowania modułów jądra, czasy rozpoczęcia i zakończenia zostaną zapisane w plikach w katalogu domowym



```
vraniu@ideapad: ~  
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/psnap.ko  
LD [M] net/802/stp.ko  
LD [M] net/8021q/8021q.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/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-5.18.2# cat ~/modulesstart  
Mon Jun 6 19:43:47 CEST 2022  
root@slack:/usr/src/linux-5.18.2# cat ~/modulesend  
Mon Jun 6 19:50:53 CEST 2022  
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 10: Zbudowanie modułów jądra trwało około 7 minut.

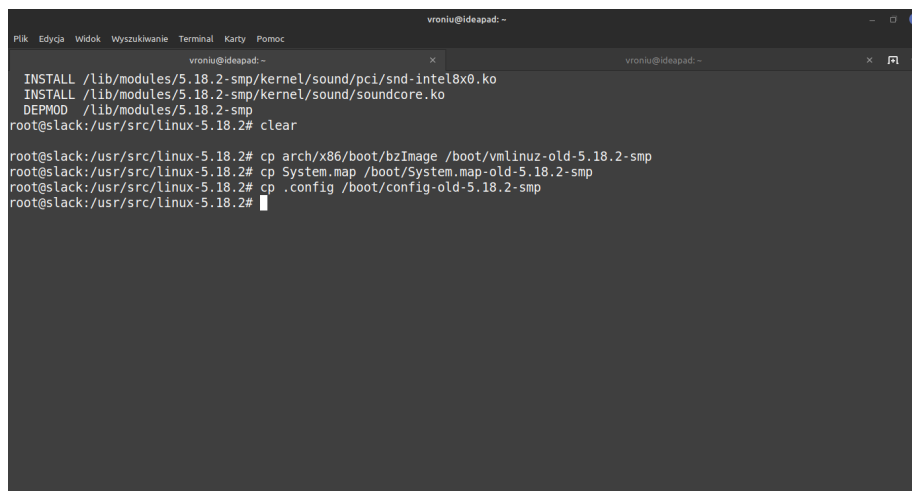
Teraz pora zainstalować moduły - do tego służy komenda `make modules_install`.



```
vraniu@ideapad: ~  
vraniu@ideapad: ~  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/powercap/intel_rapl_common.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/powercap/intel_rapl_msr.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/thermal/intel/intel_powerclamp.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/sysfb_ops.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/sysfillrect.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/sysimgblt.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/virt/vboxguest/vboxguest.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/garp.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/mrp.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/p8022.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/psnap.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/stp.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/8021q/8021q.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/ipv6/ipv6.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/llc/llc.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/rfkill/rfkill.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/wireless/cfg80211.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/ac97_bus.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/core/snd-pcm.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/core/snd-timer.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/core/snd.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/pci/ac97/snd-ac97-codec.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/pci/snd-intel8x0.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/soundcore.ko  
DEPMOD /lib/modules/5.18.2-smp  
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 11: Wynik komendy `make -j2 modules_install`

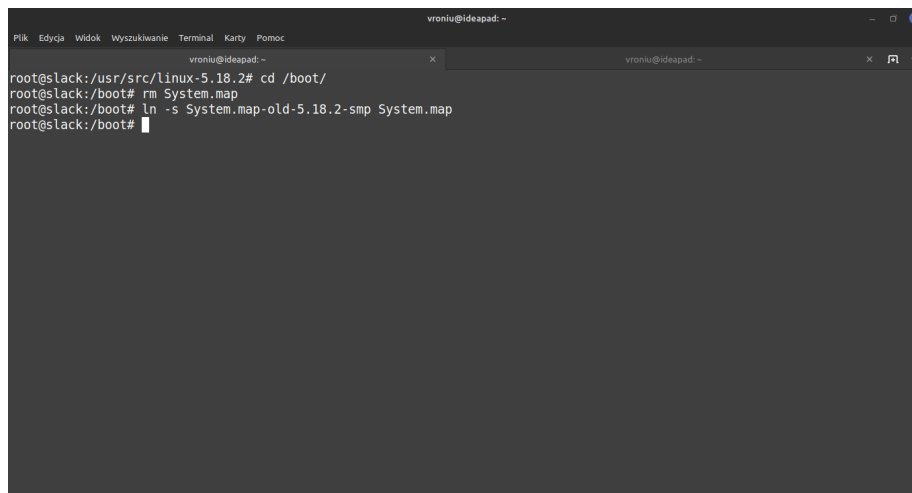
Proces kompilacji zakończony. Teraz należy przekopiować pliki nowego kernela do katalogu **boot**: obraz jądra, tablicę symboli oraz plik konfiguracyjny. Skorzystam z dobrze znanej komendy **cp**. Skopiowane pliki będą zawierały w nazwie **old-5.18.2** - wersję jądra oraz informacje, że są to pliki utworzone poprzez kompilację starą metodą.



```
vraniu@ideapad: ~  
vraniu@ideapad: ~  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/pci/snd-intel8x0.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/soundcore.ko  
DEPMOD /lib/modules/5.18.2-smp  
root@slack:/usr/src/linux-5.18.2# clear  
  
root@slack:/usr/src/linux-5.18.2# cp arch/x86/boot/bzImage /boot/vmlinuz-old-5.18.2-smp  
root@slack:/usr/src/linux-5.18.2# cp System.map /boot/System.map-old-5.18.2-smp  
root@slack:/usr/src/linux-5.18.2# cp .config /boot/config-old-5.18.2-smp  
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 12: Kopiowanie plików

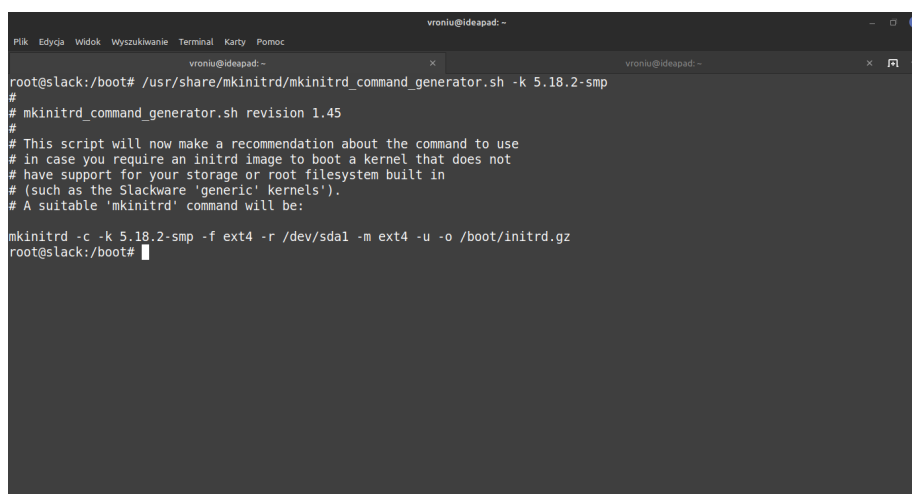
Teraz przechodzimy do wspomnianego wcześniej katalogu **boot**, usuwamy starą tablicę symboli i zastępujemy je linkiem symbolicznym do skopiowanej wcześniej tablicy.



```
vraniu@ideapad: ~  
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc  
vraniu@ideapad: ~  
root@slack:/usr/src/linux-5.18.2# cd /boot/  
root@slack:/boot# rm System.map  
root@slack:/boot# ln -s System.map-old-5.18.2-smp System.map  
root@slack:/boot#
```

Rysunek 13: Zastąpienie poprzedniej tablicy podwiązaniem do nowej

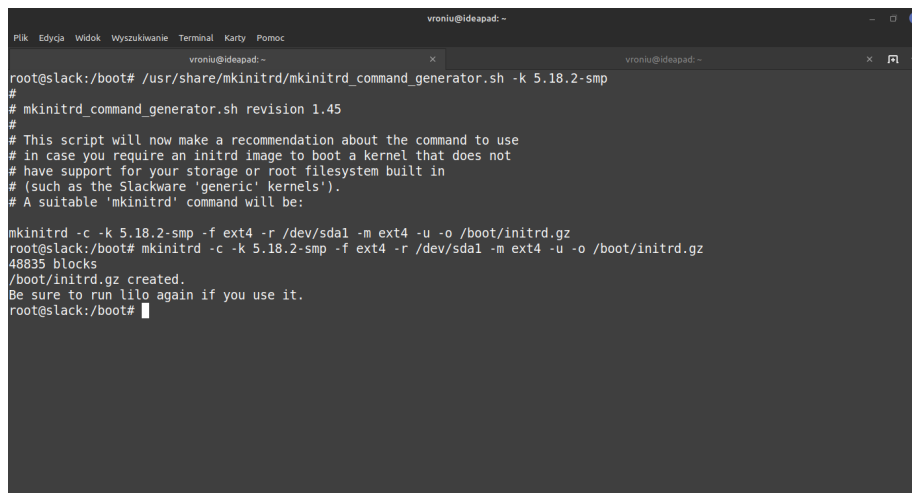
Następnie skorzystam z narzędzia, które wygeneruje mi komendę tworzącą dysk RAM. Ważna jest opcja **-k 5.18.2-smp**, która sygnalizuje wersję jądra.



```
vraniu@ideapad: ~  
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc  
vraniu@ideapad: ~  
root@slack:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.2-smp  
#  
# mkinitrd_command_generator.sh revision 1.45  
#  
# This script will now make a recommendation about the command to use  
# in case you require an initrd image to boot a kernel that does not  
# have support for your storage or root filesystem built in  
# (such as the Slackware 'generic' kernels').  
# A suitable 'mkinitrd' command will be:  
mkinitrd -c -k 5.18.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz  
root@slack:/boot#
```

Rysunek 14: Wygenerowana komenda

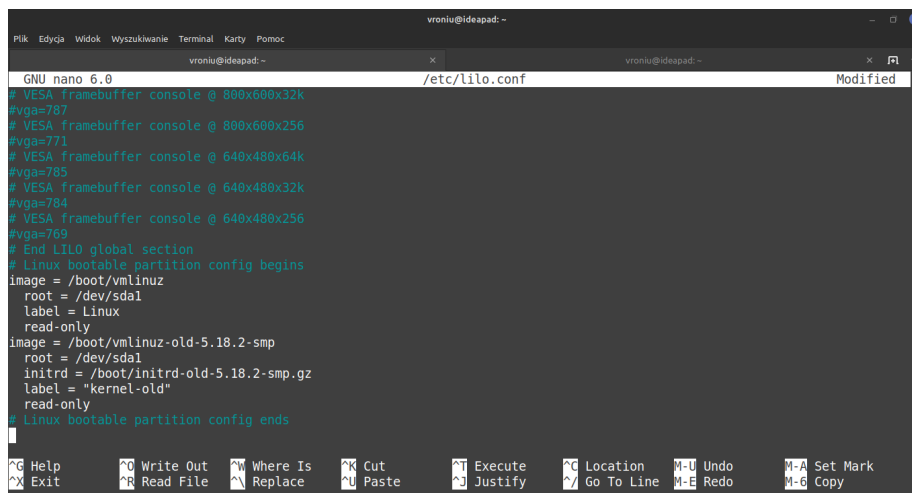
Przekopiuję wygenerowaną komendę i uruchamiam.



```
vrniu@ideapad: ~  
root@slack:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.2-smp  
#  
# mkinitrd_command_generator.sh revision 1.45  
#  
# This script will now make a recommendation about the command to use  
# in case you require an initrd image to boot a kernel that does not  
# have support for your storage or root filesystem built in  
# (such as the Slackware 'generic' kernels').  
# A suitable 'mkinitrd' command will be:  
  
mkinitrd -c -k 5.18.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz  
root@slack:/boot# mkinitrd -c -k 5.18.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz  
48835 blocks  
/boot/initrd.gz created.  
Be sure to run lilo again if you use it.  
root@slack:/boot#
```

Rysunek 15: Wynik wygenerowanej komendy - informacja o utworzonym pliku oraz przypomnienie o skonfigurowaniu lilo

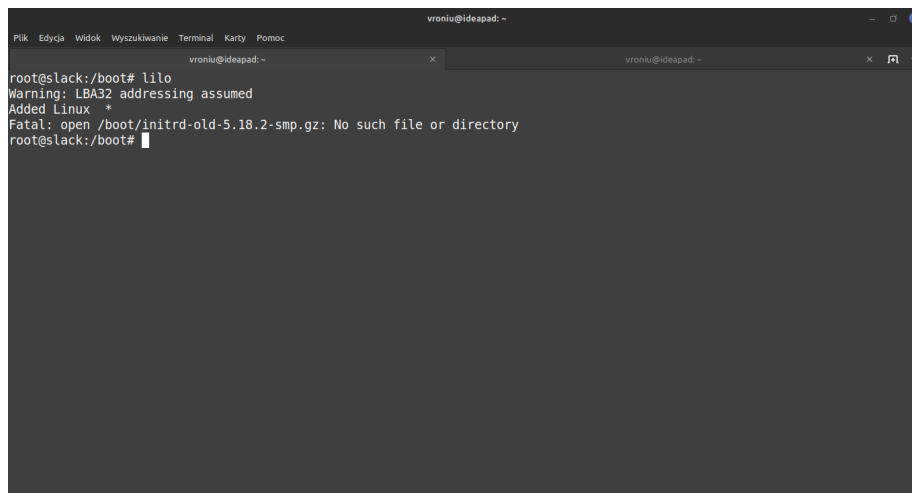
Pora ustawić bootloader lilo. Skorzystam z edytora **nano** i dodam następujący wpis w pliku **/etc/lilo.conf**, który powinien uruchamiać system z nowo skompilowanym kerneliem.



```
GNU nano 6.0 /etc/lilo.conf Modified  
# 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/sda1  
  label = Linux  
  read-only  
image = /boot/vmlinuz-old-5.18.2-smp  
  root = /dev/sda1  
  initrd = /boot/initrd-old-5.18.2-smp.gz  
  label = "kernel-old"  
  read-only  
# Linux bootable partition config ends
```

Rysunek 16: Wpis powinien być dodany pomiędzy sekcjami **Linux bootable partition config begins** i **Linux bootable partition config ends**

Zapisuję zmiany i wywołuję komendę **lilo**.

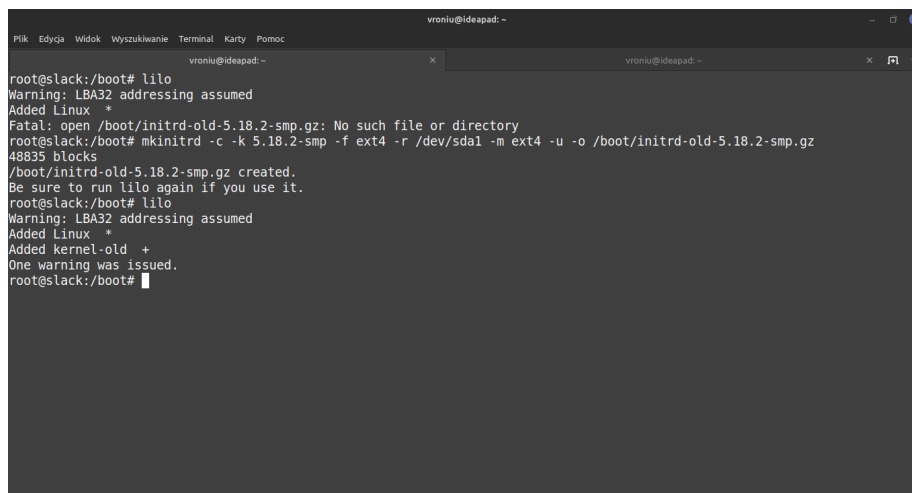
A screenshot of a terminal window titled 'vroniu@ideapad: ~'. The terminal shows the following output: root@slack:/boot# lilo, Warning: LBA32 addressing assumed, Added Linux *, Fatal: open /boot/initrd-old-5.18.2-smp.gz: No such file or directory, and root@slack:/boot#. The error message indicates that the file /boot/initrd-old-5.18.2-smp.gz does not exist.

```
vroniu@ideapad: ~
Plik  Edycja  Widok  Wyszukiwanie  Terminal  Karty  Pomoc

vroniu@ideapad: ~
root@slack:/boot# lilo
Warning: LBA32 addressing assumed
Added Linux *
Fatal: open /boot/initrd-old-5.18.2-smp.gz: No such file or directory
root@slack:/boot#
```

Rysunek 17: Błąd lilo.

Niestety popełniłem błąd podczas wywoływania wygenerowanej komendy do utworzenia dysku RAM - nie zmodyfikowałem jej i został napisany plik **/boot/initrd.gz**. Poprawiłem komendę, aby utworzyła plik o odpowiedniej nazwie i wywołałem lilo znowu.

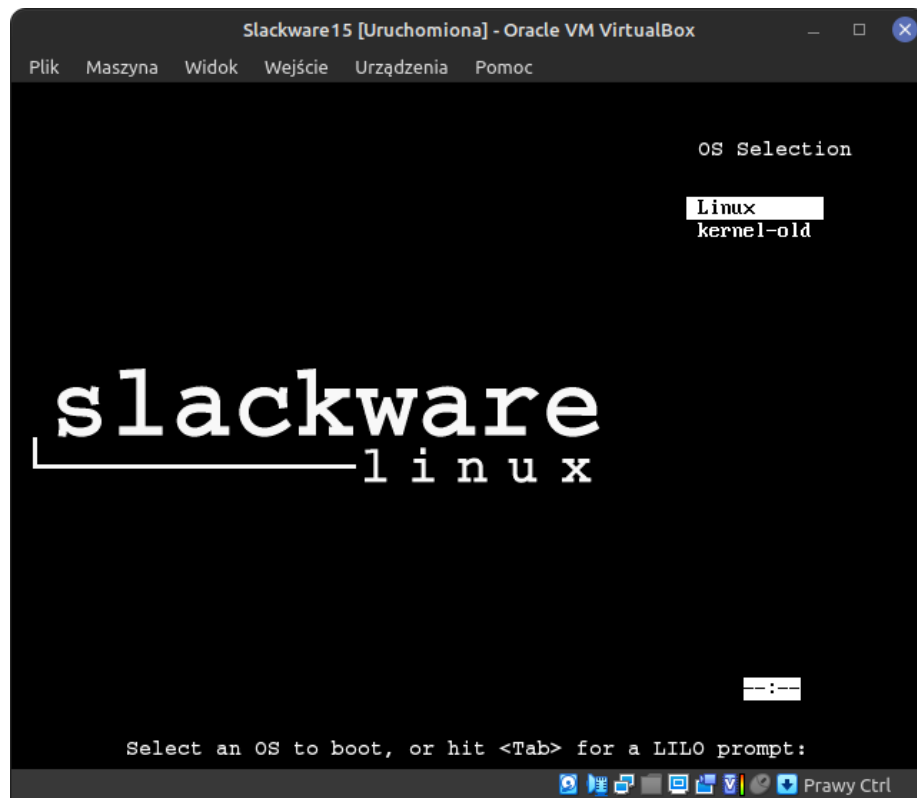
A screenshot of a terminal window titled 'vroniu@ideapad: ~'. The terminal shows the following output: root@slack:/boot# lilo, Warning: LBA32 addressing assumed, Added Linux *, Fatal: open /boot/initrd-old-5.18.2-smp.gz: No such file or directory, root@slack:/boot# mkinitrd -c -k 5.18.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-old-5.18.2-smp.gz 48835 blocks, /boot/initrd-old-5.18.2-smp.gz created, Be sure to run lilo again if you use it., root@slack:/boot# lilo, Warning: LBA32 addressing assumed, Added Linux *, Added kernel-old +, One warning was issued., and root@slack:/boot#. The output shows that the file was successfully created and lilo was run again without further errors.

```
vroniu@ideapad: ~
Plik  Edycja  Widok  Wyszukiwanie  Terminal  Karty  Pomoc

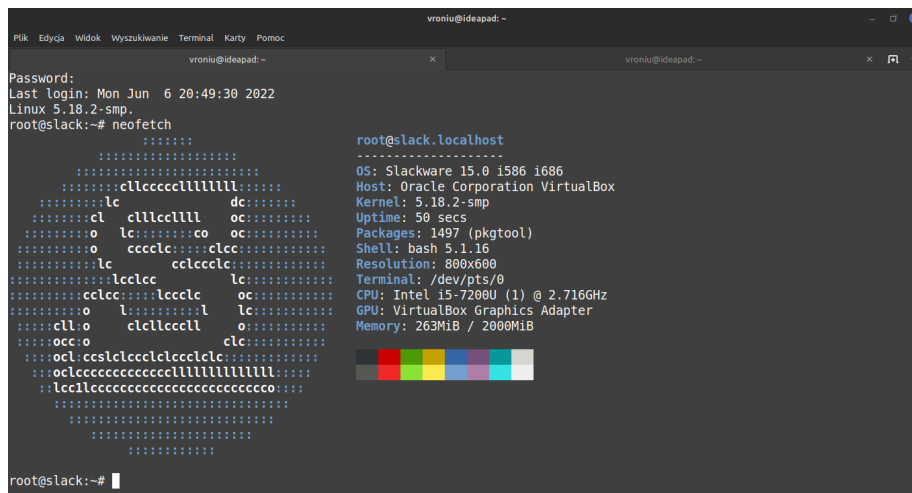
vroniu@ideapad: ~
root@slack:/boot# lilo
Warning: LBA32 addressing assumed
Added Linux *
Fatal: open /boot/initrd-old-5.18.2-smp.gz: No such file or directory
root@slack:/boot# mkinitrd -c -k 5.18.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-old-5.18.2-smp.gz 48835 blocks
/boot/initrd-old-5.18.2-smp.gz created.
Be sure to run lilo again if you use it.
root@slack:/boot# lilo
Warning: LBA32 addressing assumed
Added Linux *
Added kernel-old +
One warning was issued.
root@slack:/boot#
```

Rysunek 18: Poprawny wynik lilo.

Pora na restart wirtualnej maszyny i sprawdzenie, czy wpis pojawił się w lilo.



Rysunek 19: Wpis pojawił się w bootloaderze

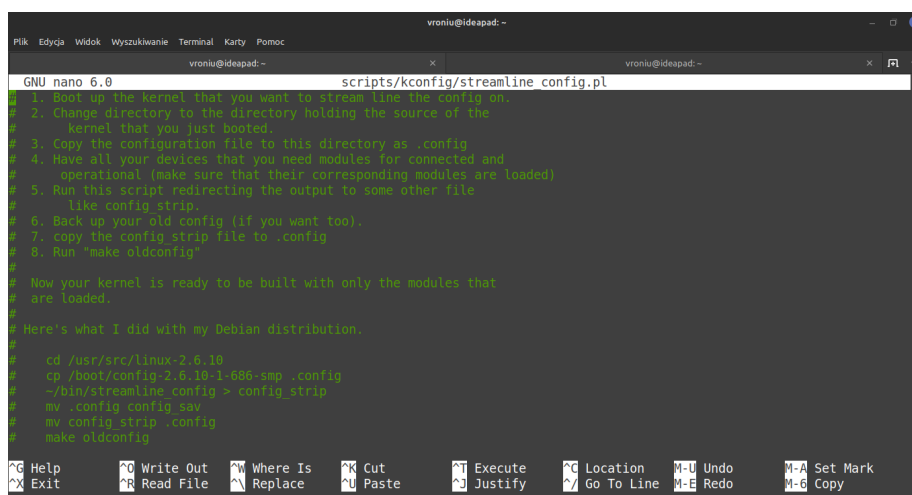


Rysunek 20: Uruchomiony system - widać nową wersję jądra

Kompilacja jądra z użyciem starej metody zakończyła się sukcesem.

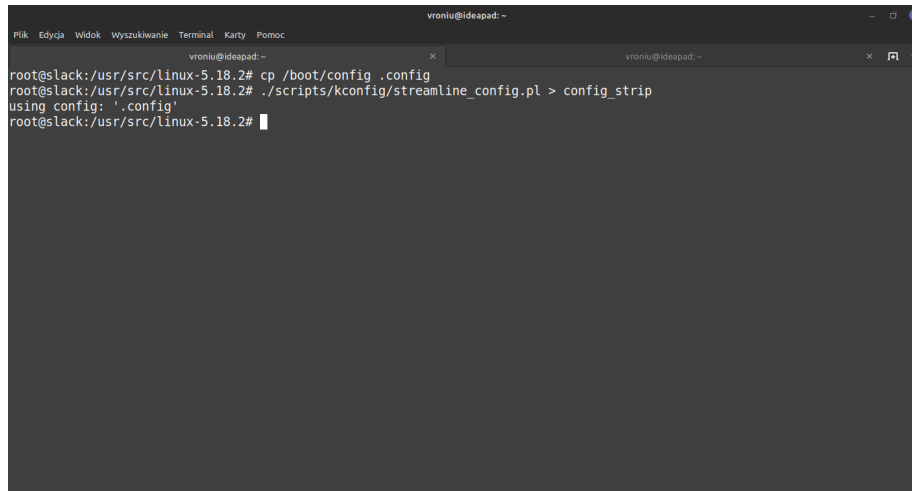
3 Kompilacja - metoda nowa

Zaczynam tak jak w poprzedniej metodzie - rozpakowuję archiwum od nowa. W pliku **scripts/kconfig/streamline_config.pl** znajdują się instrukcje do przeprowadzenia procesu konfiguracji zgodnie z nową metodą.



Rysunek 21: Instrukcje zawarte w skrypcie

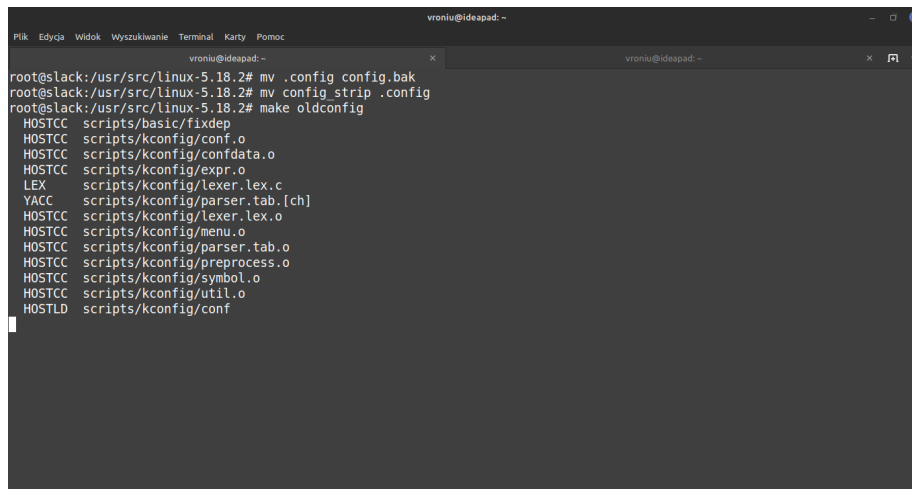
Zgodnie z instrukcjami przekopiuję plik `/boot/config` i uruchamiam skrypt, przekierowując jego wyjście do pliku `config_strip`.



```
vraniu@ideapad: ~  
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc  
vraniu@ideapad: ~  
root@slack:/usr/src/linux-5.18.2# cp /boot/config .config  
root@slack:/usr/src/linux-5.18.2# ./scripts/kconfig/streamline_config.pl > config_strip  
using config: '.config'  
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 22: Uruchomienie skryptu

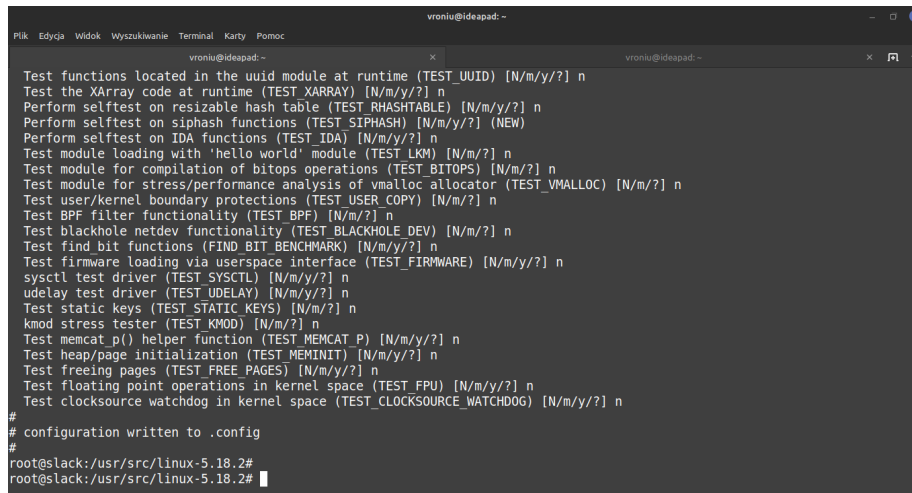
Teraz podmieniam zastępuję plik `.config` nowo utworzonym plikiem `config_strip` (tworząc przy okazji kopię zapasową) i uruchamiam polecenie `make oldconfig`



```
vraniu@ideapad: ~  
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc  
vraniu@ideapad: ~  
root@slack:/usr/src/linux-5.18.2# mv .config config.bak  
root@slack:/usr/src/linux-5.18.2# mv config_strip .config  
root@slack:/usr/src/linux-5.18.2# make oldconfig  
HOSTCC scripts/basic/fixdep  
HOSTCC scripts/kconfig/conf.o  
HOSTCC scripts/kconfig/confdata.o  
HOSTCC scripts/kconfig/expr.o  
LEX scripts/kconfig/lexer.lex.c  
YACC scripts/kconfig/parser.tab.[ch]  
HOSTCC scripts/kconfig/lexer.lex.o  
HOSTCC scripts/kconfig/menu.o  
HOSTCC scripts/kconfig/parser.tab.o  
HOSTCC scripts/kconfig/preprocess.o  
HOSTCC scripts/kconfig/symbol.o  
HOSTCC scripts/kconfig/util.o  
HOSTLD scripts/kconfig/conf
```

Rysunek 23: Zamiana plików i uruchomienie komendy

Podobnie jak w poprzedniej metodzie, klikam enter w przypadku komunikatów o konfiguracji.



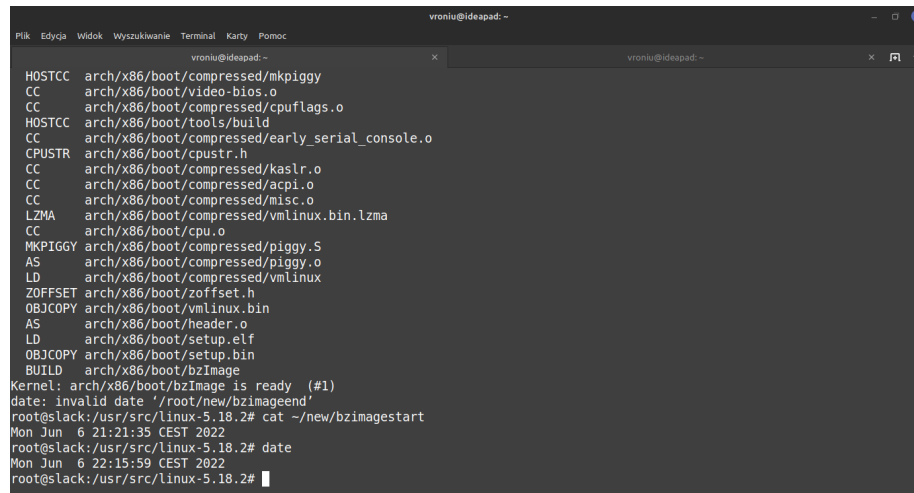
```
Test functions located in the uuid module at runtime (TEST_UUID) [N/m/y/?] n
Test the XArray code at runtime (TEST_XARRAY) [N/m/y/?] n
Perform selftest on resizable hash table (TEST_RHASHTABLE) [N/m/y/?] n
Perform selftest on siphash functions (TEST_SIPHASH) [N/m/y/?] (NEW)
Perform selftest on IDA 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_VMLLOC) [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-5.18.2#
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 24: Wynik komendy `make oldconfig`

Teraz analogicznie jak w poprzedniej metodzie - najpierw kompilacja jądra obrazu z odpowiednim parametrem `-j` oraz z zmierzeniem czasu - wykorzystana komenda:

```
date > /new/bzimagestart && make -j2 bzImage && date
/new/bzimageend
```

Niestety tutaj popełniłem mały błąd i zapomniałem dać przekierowania przy drugiej komendzie `date`. Wobec tego nie zapisało się, jak długo trwał proces, ale trwał on bardzo podobnie czasowo jak w przypadku starej metody.

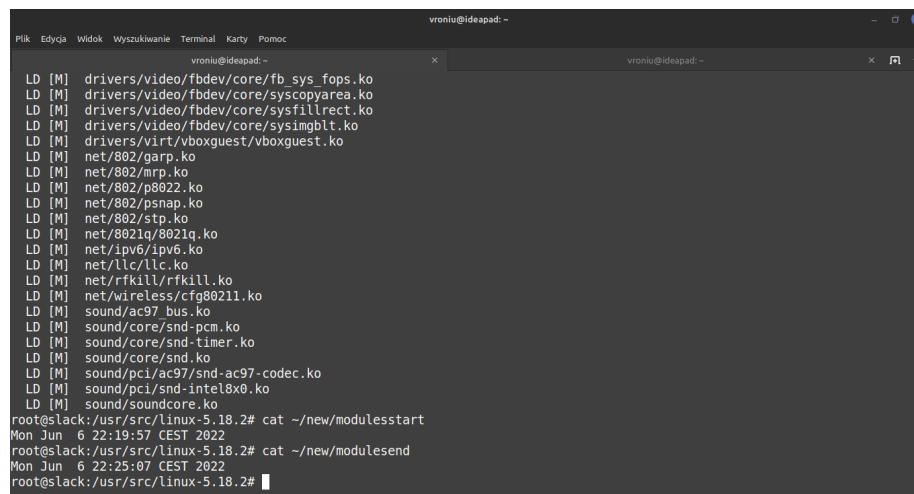


```
vrioniu@ideapad: ~  
HOSTCC arch/x86/boot/compressed/mkpiggy  
CC arch/x86/boot/video-bios.o  
CC arch/x86/boot/compressed/cpuflags.o  
HOSTCC arch/x86/boot/tools/build  
CC arch/x86/boot/compressed/early_serial_console.o  
CPUPSTR arch/x86/boot/cpustr.h  
CC arch/x86/boot/compressed/kaslr.o  
CC arch/x86/boot/compressed/acpi.o  
CC arch/x86/boot/compressed/misc.o  
LZMA arch/x86/boot/compressed/vmlinux.bin.lzma  
CC arch/x86/boot/cpu.o  
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)  
date: invalid date '/root/new/bzimageend'  
root@slack:/usr/src/linux-5.18.2# cat ~/new/bzimagestart  
Mon Jun 6 21:21:35 CEST 2022  
root@slack:/usr/src/linux-5.18.2# date  
Mon Jun 6 22:15:59 CEST 2022  
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 25: Koniec kompilacji obrazu jądra - podobnie, jak wcześniej, trwało to około 54 minuty.

Czas na kompilację modułów - wykorzystana komenda:

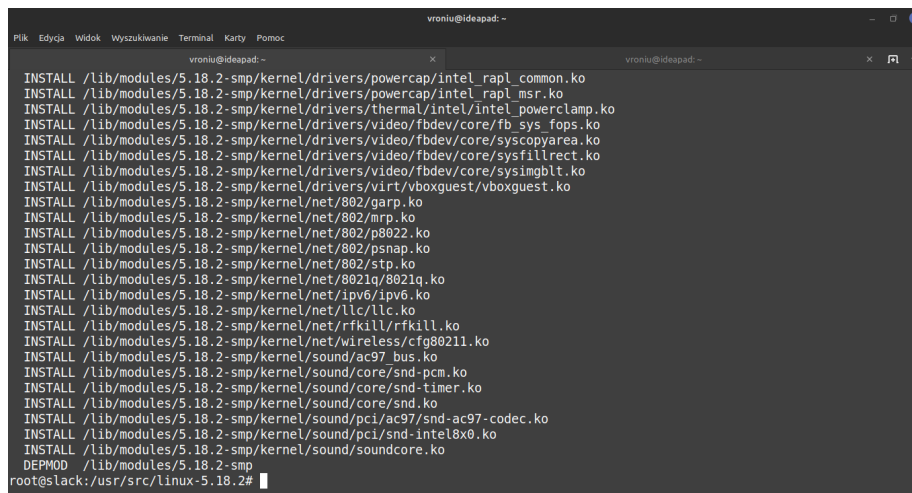
```
date > /new/modulesstart && make -j2 modules && date >  
/new/modulesend
```



```
vrioniu@ideapad: ~  
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/psnap.ko  
LD [M] net/802/stp.ko  
LD [M] net/8021q/8021q.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/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-5.18.2# cat ~/new/modulesstart  
Mon Jun 6 22:19:57 CEST 2022  
root@slack:/usr/src/linux-5.18.2# cat ~/new/modulesend  
Mon Jun 6 22:25:07 CEST 2022  
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 26: Wynik kompilacji modułów - wykonanie komendy zajęło około 6 minut

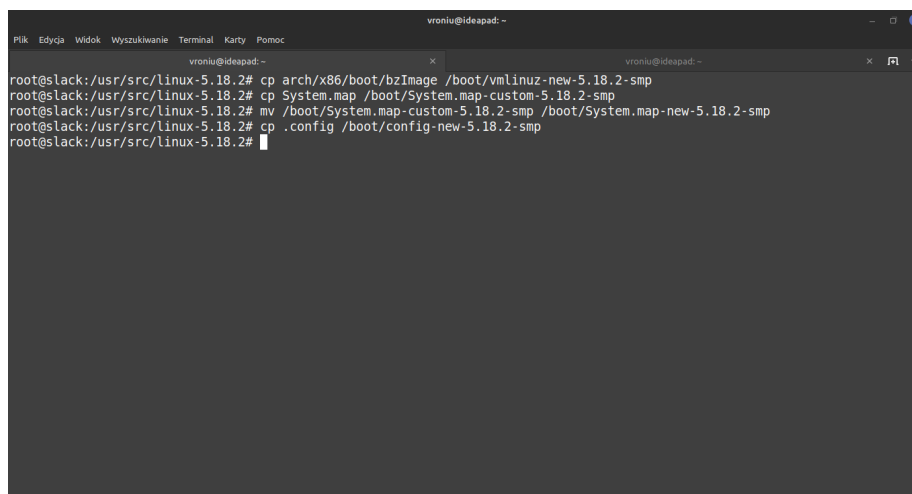
Instalacja modułów - komenda `make modules_install`



```
vrioni@ideapad: ~  
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc  
vrioni@ideapad: ~  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/powercap/intel_rapl_common.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/powercap/intel_rapl_msr.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/thermal/intel/intel_powerclamp.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/fb_sys_fops.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/syscopyarea.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/sysfillrect.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/sysimgblt.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/drivers/virt/vboxguest/vboxguest.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/garp.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/mrp.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/p8022.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/psnap.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/802/stp.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/8021q/8021q.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/ipv6/ipv6.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/lc/lc.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/rfkill/rfkill.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/net/wireless/cfg80211.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/ac97/bus.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/core/snd-pcm.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/core/snd-timer.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/core/snd.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/pci/ac97/snd-ac97-codec.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/pci/snd-intel8x0.ko  
INSTALL /lib/modules/5.18.2-smp/kernel/sound/soundcore.ko  
DEPMOD /lib/modules/5.18.2-smp  
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 27: Wynik instalacji modułów

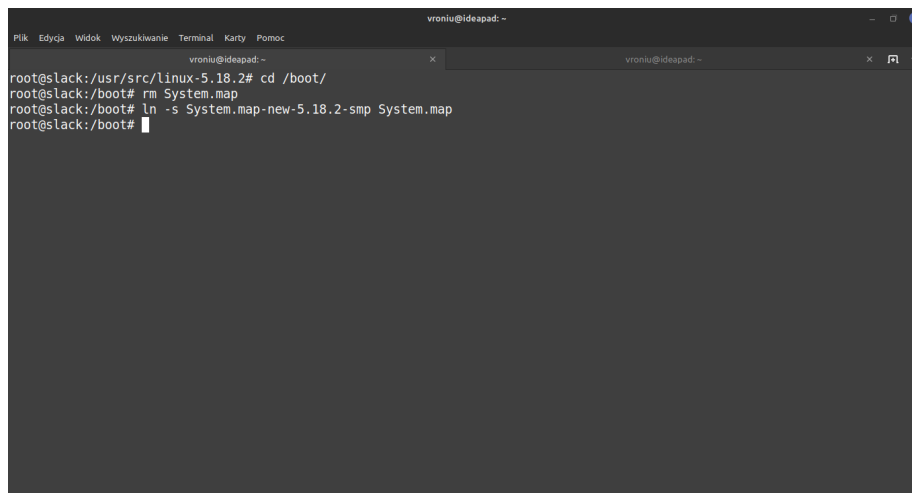
Teraz przekopiuję odpowiednie pliki do folderu **/boot** - wkradła mi się mała literówka przy kopiowaniu tablicy, ale ją poprawiłem.



```
vrioni@ideapad: ~  
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc  
vrioni@ideapad: ~  
root@slack:/usr/src/linux-5.18.2# cp arch/x86/boot/bzImage /boot/vmlinuz-new-5.18.2-smp  
root@slack:/usr/src/linux-5.18.2# cp System.map /boot/System.map-custom-5.18.2-smp  
root@slack:/usr/src/linux-5.18.2# mv /boot/System.map-custom-5.18.2-smp /boot/System.map-new-5.18.2-smp  
root@slack:/usr/src/linux-5.18.2# cp .config /boot/config-new-5.18.2-smp  
root@slack:/usr/src/linux-5.18.2#
```

Rysunek 28: Kopiowanie plików

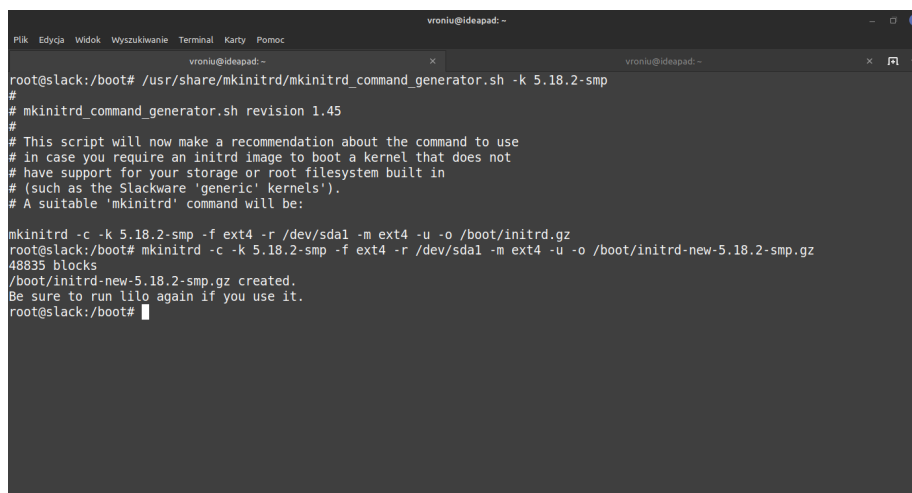
Teraz przechodzę do katalogu **/boot** i zastępuję ją linkiem do nowej.



```
vraniu@ideapad: ~  
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc  
vraniu@ideapad: ~  
root@slack:/usr/src/linux-5.18.2# cd /boot/  
root@slack:/boot# rm System.map  
root@slack:/boot# ln -s System.map-new-5.18.2-smp System.map  
root@slack:/boot#
```

Rysunek 29: Podmiana tablicy na nową

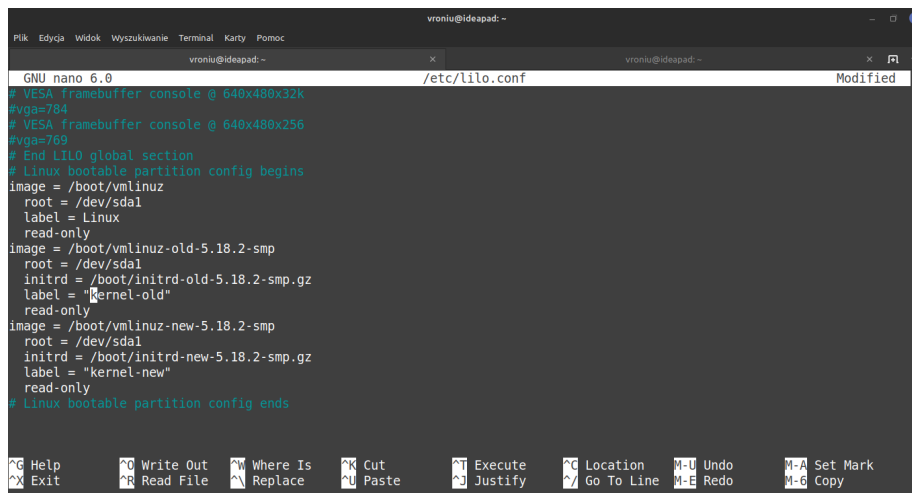
Następne w kolejności jest wygenerowanie komendy do utworzenia dysku RAM i wykonanie jej - tym razem pamiętałem, żeby zmienić nazwę wyjściowego pliku.



```
vraniu@ideapad: ~  
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc  
vraniu@ideapad: ~  
root@slack:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.2-smp  
#  
# mkinitrd_command_generator.sh revision 1.45  
#  
# This script will now make a recommendation about the command to use  
# in case you require an initrd image to boot a kernel that does not  
# have support for your storage or root filesystem built in  
# (such as the Slackware 'generic' kernels').  
# A suitable 'mkinitrd' command will be:  
  
mkinitrd -c -k 5.18.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz  
root@slack:/boot# mkinitrd -c -k 5.18.2-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-new-5.18.2-smp.gz  
48835 blocks  
/boot/initrd-new-5.18.2-smp.gz created.  
Be sure to run lilo again if you use it.  
root@slack:/boot#
```

Rysunek 30: Tworzenie dysku RAM

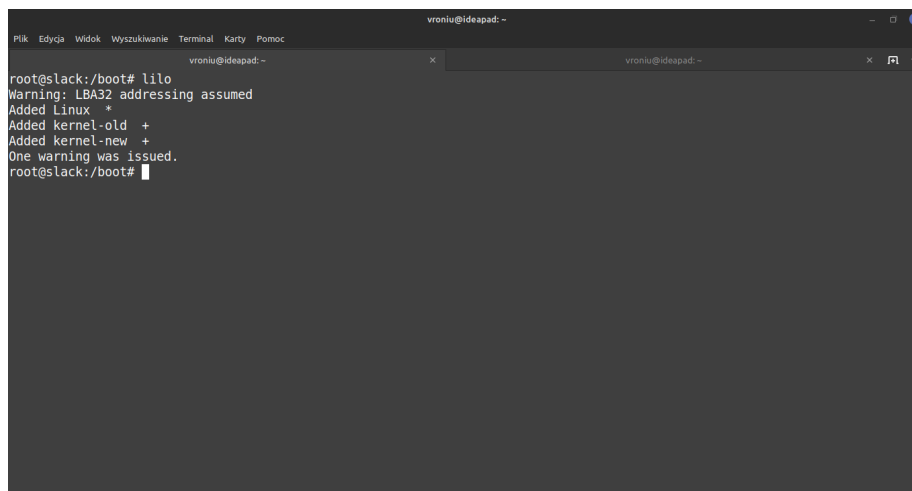
Pozostało dodać wpis do pliku `/etc/lilo.conf`...



```
GNU nano 6.0 /etc/lilo.conf Modified
# VESA framebuffer console @ 640x480x32K
vga=784
# VESA framebuffer console @ 640x480x256
vga=769
# End LILO global section
# Linux bootable partition config begins
image = /boot/vmlinuz
  root = /dev/sda1
  label = Linux
  read-only
image = /boot/vmlinuz-old-5.18.2-smp
  root = /dev/sda1
  initrd = /boot/initrd-old-5.18.2-smp.gz
  label = "kernel-old"
  read-only
image = /boot/vmlinuz-new-5.18.2-smp
  root = /dev/sda1
  initrd = /boot/initrd-new-5.18.2-smp.gz
  label = "kernel-new"
  read-only
# Linux bootable partition config ends
```

Rysunek 31: Wpis w pliku konfiguracyjnym lilo

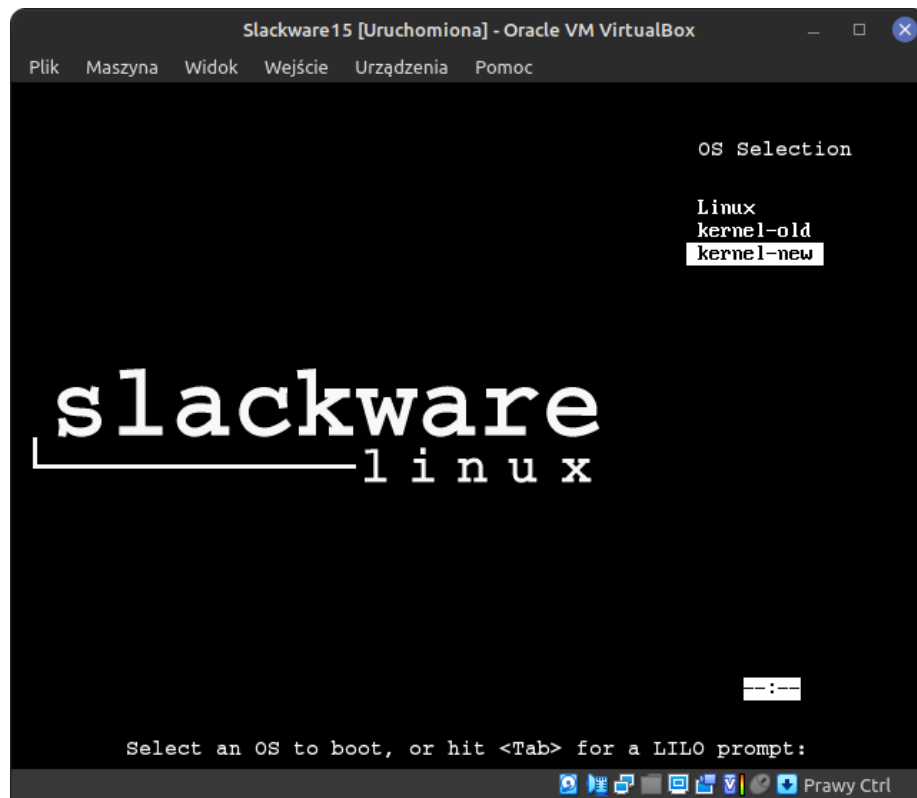
...i uruchomić komendę `lilo`.



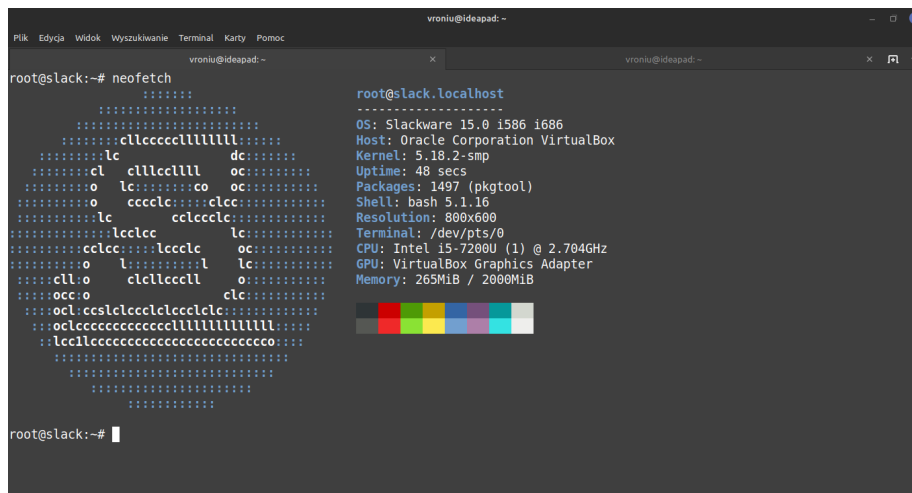
```
root@slack:/boot# lilo
Warning: LBA32 addressing assumed
Added linux +
Added kernel-old +
Added kernel-new +
One warning was issued.
root@slack:/boot#
```

Rysunek 32: Wynik wywołania komendy `lilo`

Czas na restart systemu.



Rysunek 33: W boootloaderze pojawiła się nowo dodana przez nas opcja



```
root@slack:~# neofetch

root@slack.localhost
OS: Slackware 15.0 i586 i686
Host: Oracle Corporation VirtualBox
Kernel: 5.18.2-5mp
Uptime: 48 secs
Packages: 1497 (pkgtool)
Shell: bash 5.1.16
Resolution: 800x600
Terminal: /dev/pts/0
CPU: Intel i5-7200U (1) @ 2.704GHz
GPU: VirtualBox Graphics Adapter
Memory: 265MiB / 2000MiB
```

Rysunek 34: System ładuje się poprawnie. Kernel systemu został zaaktualizowany poprawnie.

Proces kompilacji i instalacji jądra nową metodą zakończył się sukcesem.

4 Podsumowanie

Kompilacja jądra z wykorzystaniem obydwu metod zakończyła się sukcesem. Dość zaskakujące jest, jak podobne są obie metody - zarówno w sposobie wykonywania kompilacji z punktu widzenia użytkownika, jak i w czasie kompilacji (różnice czasowe w moim przypadku były minimalne). Jeżeli chodzi o moje odczucia, to według mnie obie te metody są dość przyjazne dla użytkownika - na szczęście nie miałem żadnych poważnych problemów ani ze starą, ani z nową metodą. Mam natomiast zastrzeżenia co do etapu tworzenia pliku konfiguracyjnego (komendy `make localmodconfig`/`make oldconfig`) - pytania zadawane przez tą komendę są bardzo zaawansowane i ciężkie do zrozumienia nawet dla dość zaawansowanych użytkowników. Na szczęście 'przeklikanie' enter rozwiązuje ten problem, ale w tym przypadku przydała by się możliwość automatycznego ustawienia wszystkich ustawień na domyślne.

Mój finalny werdykt - obie metody są okej, ale skłaniał bym się ku nowszej ze względu na przejrzystą instrukcję zawartą w pliku `streamline_config.pl`.