

Kompilacja jądra Linuxa

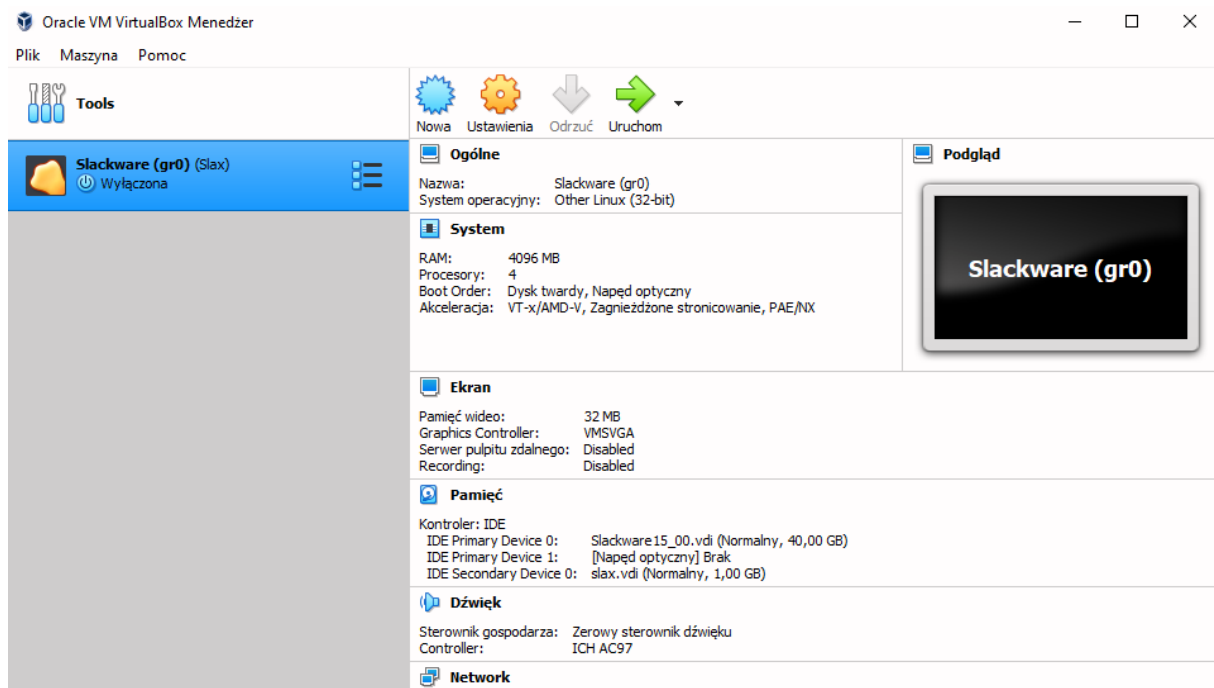
Krystian Trześniak

296505

16.06 2022

1. Przygotowanie do pracy

Zanim przystąpię do wykonania zadania sprawdzam czy na pewno używam wystarczającej ilości pamięci RAM i czy wykorzystuję wszystkie 4 rdzenie procesora.



Rysunek 1 Przygotowanie maszyny

Początkowo miałem mały problem z zalogowaniem się na koncie roota (zapomniałem hasła). Po kilku próbach i nerwowych restartach maszyny przypomniałem sobie hasło i mogłem przejść do kolejnych etapów zadania.


```

root@slack:~# wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.5.tar.xz
--2022-06-16 20:44:29-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.5.tar.xz
Translacja cdn.kernel.org (cdn.kernel.org)... 151.101.13.176, 2a04:4e42:3::432
Początek połączenia z cdn.kernel.org (cdn.kernel.org)[151.101.13.176]:443... połączono.
Przekazywanie 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  27,4MB/s   w 4,5s

2022-06-16 20:44:34 (27,5 MB/s) - zapisano `linux-5.18.5.tar.xz' [129825940/129825940]

root@slack:~#

```

Rysunek 4 Wynik polecenia wget

Archiwum postanowiłem rozpakować przy pomocy polecenia tar, uprzednio przenosząc je do katalogu /usr/src.

```

root@slack:~# ls
linux-5.18.5.tar.xz
root@slack:~# mv linux-5.18.5.tar.xz /usr/src
root@slack:~# ls
root@slack:~# cd usr/src/
-bash: cd: usr/src/: Nie ma takiego pliku ani katalogu
root@slack:~# cd usr/src
-bash: cd: usr/src: Nie ma takiego pliku ani katalogu
root@slack:~# cd /usr/src
root@slack:/usr/src# ls
linux-5.18.5.tar.xz
root@slack:/usr/src# tar -xpf linux-5.18.5.tar.xz

root@slack:/usr/src#
root@slack:/usr/src# ls
linux-5.18.5/ linux-5.18.5.tar.xz
root@slack:/usr/src# _

```

Rysunek 5 Przeniesienie archiwum do katalogu /usr/src i wypakowanie

Koniec przygotowania i początek zabawy, w następnych krokach będę realizował kompilację jądra.

2. Kompilacja jądra linux starą metodą

Na początku należy przejść do folderu z rozpakowanymi plikami. Konfigurację rozpocznę od skopiowania konfiguracji starego jądra do pliku .config.

```

root@slack:/usr/src# cd linux-5.18.5
root@slack:/usr/src/linux-5.18.5# zcat /proc/config.gz > .config
root@slack:/usr/src/linux-5.18.5# ls | grep .config
Kconfig
root@slack:/usr/src/linux-5.18.5# ls -la | grep .config
-rw-rw-r-- 1 root root 59 cze 16 13:32 .cocciconfig
-rw-r--r-- 1 root root 237798 cze 16 20:56 .config
-rw-rw-r-- 1 root root 555 cze 16 13:32 Kconfig
root@slack:/usr/src/linux-5.18.5# _

```

Rysunek 6 Kopia starej konfiguracji

Dodatkowo upewniam się czy wszystko się dobrze skopiowało używając polecenia grep. Następnym krokiem będzie wywołanie komendy make localmodconfig aby przygotować plik konfiguracyjny.

```

Test functions located in the hexdump module at runtime (TEST_HEXDUMP) [N/m/y/?] n
Test string functions at runtime (STRING_SELFTEST) [N/m/y/?] n
Test functions located in the string_helpers module at runtime (TEST_STRING_HELPERS) [N/m/y/?] n
Test strscpy*() family of functions at runtime (TEST_STRSCPY) [N/m/y/?] n
Test kstrt*() family of functions at runtime (TEST_KSTRTX) [N/m/y/?] n
Test printf() family of functions at runtime (TEST_PRINTF) [N/m/y/?] n
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_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 umalloc allocator (TEST_UMALLOC) [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# _

```

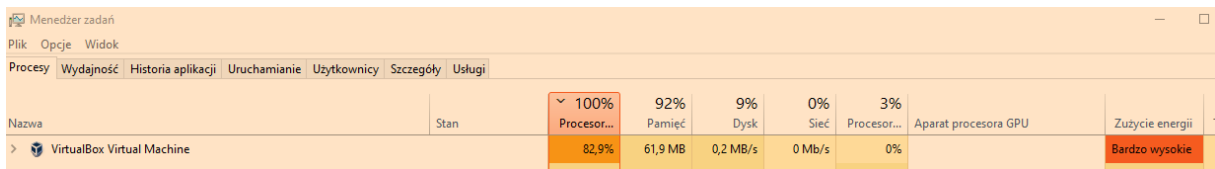
Rysunek 7 Wynik polecenia make localmodconfig

Na rysunku 7. widnieje wynik polecenia make localmodconfig. Ten krok wymagał ode mnie rozpedzenie przycisku enter do prędkości nieosiągalnych dla przeciętnego użytkownika Windows. Sekwencja kliknięć enter zaprowadziła mnie do zapisania konfiguracji w pliku .config.

Moja maszyna przy przygotowaniu dostała 4 rdzenie, więc przy procesie kompilacji użyję komendy `make bzImage` z parametrem `-j4`. Być może to wpłynie pozytywnie na prędkość wykonania operacji.

```
root@slack:/usr/src/linux-5.18.5# make -j4 bzImage_
```

Rysunek 8 Polecenie kompilujące



Nazwa	Stan	Procesor...	Pamięć	Dysk	Sieć	Procesor...	Aparat procesora GPU	Zużycie energii
VirtualBox Virtual Machine		100% 82,9%	92% 61,9 MB	9% 0,2 MB/s	0% 0 Mb/s	3% 0%		Bardzo wysokie

Rysunek 9 Zużycie procesora podczas kompilacji

Proces kompilacji jest niestety czasochłonny i z uwagi na wysokie zużycie procesora mój edytor tekstowy zaczął przycinać. Udaję się więc na przerwę.



Rysunek 10 Przerwa na kawę

Po ~20 minutach kompilacja dobiegła końca i mogę już przejść do kompilacji modułów.

```

CC      arch/x86/boot/string.o
AS      arch/x86/boot/compressed/head_32.o
VOFFSET arch/x86/boot/compressed/./voffset.h
CC      arch/x86/boot/tty.o
CC      arch/x86/boot/video.o
CC      arch/x86/boot/video-mode.o
CC      arch/x86/boot/version.o
CC      arch/x86/boot/video-vga.o
CC      arch/x86/boot/video-vesa.o
CC      arch/x86/boot/video-bios.o
HOSTCC  arch/x86/boot/tools/build
CC      arch/x86/boot/compressed/string.o
CC      arch/x86/boot/compressed/cmdline.o
CPUSTR  arch/x86/boot/cpustr.h
CC      arch/x86/boot/compressed/error.o
OBJCOPY arch/x86/boot/compressed/vmlinux.bin
CC      arch/x86/boot/cpu.o
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-5.18.5#
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 11 Wynik kompilacji jądra

Aby skompilować moduły użyję komendy `make -j4 modules`.

```
LD [M] drivers/gpu/drm/drm_ttm_helper.ko
LD [M] drivers/gpu/drm/umwgf/umwgf.ko
LD [M] drivers/gpu/drm/ttm/ttm.ko
LD [M] drivers/i2c/algos/i2c-algo-bit.ko
LD [M] drivers/i2c/busses/i2c-piix4.ko
LD [M] drivers/i2c/i2c-core.ko
LD [M] drivers/input/evdev.ko
LD [M] drivers/input/joydev.ko
LD [M] drivers/input/mouse/psmouse.ko
LD [M] drivers/input/serio/serio_raw.ko
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/sysfillrect.ko
LD [M] drivers/video/fbdev/core/sysimgblt.ko
LD [M] net/802/garp.ko
LD [M] net/802/mrp.ko
LD [M] drivers/virt/uboxguest/uboxguest.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/snd-intel8x0.ko
LD [M] sound/pci/ac97/snd-ac97-codec.ko
LD [M] sound/soundcore.ko
root@slack:/usr/src/linux-5.18.5# _
```

Rysunek 12 Wynik kompilacji modułów

Następnym krokiem będzie instalacja modułów, zrobię to za pomocą polecenia `make -j4 modules_install`.

```
INSTALL /lib/modules/5.18.5-smp/kernel/sound/pci/snd-intel8x0.ko
INSTALL /lib/modules/5.18.5-smp/kernel/sound/soundcore.ko
DEPMOD /lib/modules/5.18.5-smp
root@slack:/usr/src/linux-5.18.5# _
```

Rysunek 13 Wynik polecenia `make modules_install`

Następnie przekopiuję pliki jądra do katalogu boot. Oczywiście skorzystam z polecenia `cp`.

```
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 14 Kopiowanie plików nowego jądra

Kolejnym krokiem będzie przejście do katalogu boot i usunięcie starej tablicy symboli aby zastąpić ją linkiem symbolicznym do skopiowanej przed chwilą tablicy.


```
root@slack:/usr/src/linux-5.18.5# cd /boot
root@slack:/boot# rm System.map
root@slack:/boot# ln -s System.map-old-5.18.5-smp System.map
root@slack:/boot#
```

Rysunek 15 Zastąpienie tablicy linkiem symbolicznym

W kolejnym kroku wygeneruję komendę tworzącą dysk RAM z opcją -k 5.18.5-smp aby przekazać narzędziu wersję kernela.

```
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# _
```

Rysunek 16 Wygenerowanie komendy

Przepisuję komendę i klikam enter.

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

Rysunek 17 Wynik wygenerowanej komendy

Dostaję informację o tym, że powinienem skonfigurować lilo. Aby to zrobić skorzystam z edytora nano. Dodam również wpis, który będzie uruchamiać system z nowym jądrem. Wpis należy umieścić pomiędzy sekcjami Linux bootable.


```
GNU nano 6.0 /etc/lilo.conf Zmieniony
tuga = normal
# Ask for video mode at boot (time out to normal in 30s)
tuga = ask
# VESA framebuffer console @ 1024x768x64k
tuga=791
# VESA framebuffer console @ 1024x768x32k
tuga=790
# VESA framebuffer console @ 1024x768x256
tuga=773
# VESA framebuffer console @ 800x600x64k
tuga=788
# VESA framebuffer console @ 800x600x32k
tuga=787
# VESA framebuffer console @ 800x600x256
tuga=771
# VESA framebuffer console @ 640x480x64k
tuga=785
# VESA framebuffer console @ 640x480x32k
tuga=784
# VESA framebuffer console @ 640x480x256
tuga=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-old-5.18.5-smp
  root = /dev/sda1
  initrd = /boot/initrd-old-5.18.5-smp.gz
  label = "kernel-old"
  read-only
# Linux bootable partition config ends

Pomoc      ^O Zapisz    ^W Wyszukaj   ^R Wytnij    ^T Wykonaj   ^C Lokalizacja ^U Odwołaj
Wyjd       ^R Wczyt.plik ^M Zastp      ^U Wklej     ^J Wyjustuj  ^_ Do linii    ^E Odtwórz
```

Rysunek 18 /etc/lilo.conf po zmianach

Po wywołaniu komendy lilo wyskoczyło mi kilka warningów, które ignoruję. Więszym problemem natomiast okazał się bład podczas generowania komendy do tworzenia dysku RAM. Przez mój bład został nadpisany plik /boot/initrd.gz.

```
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 *
Fatal: open /boot/initrd-old-5.18.5-smp.gz: No such file or directory
root@slack:/boot#
```

Rysunek 19 Fatalny bład

Na szczucie solucja była bardzo prosta i wystarczyło poprawić komendę do tworzenia dysku RAM.

```
root@slack:/boot# mkinitrd -c -k 5.18.5-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-old-5.18.5-smp.gz
49039 bloków
/boot/initrd-old-5.18.5-smp.gz created.
Be sure to run lilo again if you use it.
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 kernel-old +
3 warnings were issued.
root@slack:/boot#
```

Rysunek 20 Solucja problemu

Ostatnim krokiem będzie ponowne uruchomienie maszyny i sprawdzenie czy wpis znajduje się w lilo.



Rysunek 21 Sukces starej metody!

To jeszcze nie koniec, należy sprawdzić czy widać nową wersję jądra. Użyj do tego ponownie komendy neofetch.


```
#!/usr/bin/env perl
# SPDX-License-Identifier: GPL-2.0
#
# Copyright 2005-2009 - Steven Rostedt
#
# It's simple enough to figure out how this works.
# If not, then you can ask me at stripconfig@goodmis.org
#
# What it does?
#
# If you have installed a Linux kernel from a distribution
# that turns on way too many modules than you need, and
# you only want the modules you use, then this program
# is perfect for you.
#
# It gives you the ability to turn off all the modules that are
# not loaded on your system.
#
# 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
```

Rysunek 23 Plik z instrukcją do kompilacji nową metodą

Kierując się wskazówkami w instrukcji przekopiuje plik /boot/config. Po tym kroku uruchamiam skrypt i kieruję jego wyjście do pliku config_strip. Na razie bez trudności.

```
root@slack:/usr/src/linux-5.18.5# cp /boot/config .config
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#
```

Rysunek 24 Pierwsze kroki z instrukcji

W następnym kroku zastępuję plik .config nowym plikiem config_strip.

```

root@slack:/usr/src/linux-5.18.5# mv .config config.bak
root@slack:/usr/src/linux-5.18.5# mv config_strip .config
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 25 Zastąpienie pliku .config

Teraz uruchamiam skrypt make oldconfig i odpalam auto-clickera na enter.

```

Min heap test (TEST_MIN_HEAP) [N/m/y/?] n
64bit/32bit division and modulo test (TEST_DIV64) [N/m/y/?] n
Self test for the backtrace code (BACKTRACE_SELF_TEST) [N/m/y/?] n
Self test for reference tracker (TEST_REF_TRACKER) [N/m/y/?] (NEW)
Red-Black tree test (RBTREE_TEST) [N/m/y/?] n
Reed-Solomon library test (REED_SOLOMON_TEST) [N/m/y/?] n
Interval tree test (INTERVAL_TREE_TEST) [N/m/y/?] n
Per cpu operations test (PERCPU_TEST) [N/m/?] n
Perform an atomic64_t self-test (ATOMIC64_SELFTEST) [Y/n/m/?] y
Self test for hardware accelerated raid6 recovery (ASYNC_RAID6_TEST) [N/m/y/?] n
Test functions located in the hexdump module at runtime (TEST_HEXDUMP) [N/m/y/?] n
Test string functions at runtime (STRING_SELFTEST) [N/m/y/?] n
Test functions located in the string_helpers module at runtime (TEST_STRING_HELPERS) [N/m/y/?] n
Test strscpy*() family of functions at runtime (TEST_STRSCPY) [N/m/y/?] n
Test kstrto*() family of functions at runtime (TEST_KSTRTOX) [N/m/y/?] n
Test printf() family of functions at runtime (TEST_PRINTF) [N/m/y/?] n
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_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 umalloc allocator (TEST_UMALLOC) [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#
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 26 Wynik polecenia make oldconfig

Nastąpił czas na kompilację jądra, zrobię to tak jak poprzednio – komendą make -j4 bzImage.

```

root@slack:/usr/src/linux-5.18.5# make -j4 bzImage
  SYNC    include/config/auto.conf.cmd
  CALL    scripts/atomic/check-atomics.sh
  CALL    scripts/checksyscalls.sh
  CHK     include/generated/compile.h
Kernel: arch/x86/boot/bzImage is ready  (#1)
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 27 Kompilacja jądra

Ku mojemu zaskoczeniu kompilacja przebiegła bardzo szybko i nie zdążyłem nawet pójść po herbatę (bo kawy na dziś już za wiele). Przystąpię więc do kompilacji modułów, używając komendy `make -j4 modules`, tak jak ostatnio.

```

root@slack:/usr/src/linux-5.18.5# make -j4 modules
  CALL    scripts/atomic/check-atomics.sh
  CALL    scripts/checksyscalls.sh
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 28 Kompilacja modułów

Tu już zaskoczenia nie było i spodziewałem się szybkiej kompilacji. Następnie należy zainstalować moduły i użyję do tego oczywiście komendy `make modules_install`.

```

INSTALL /lib/modules/5.18.5-smp/kernel/net/802/stp.ko
INSTALL /lib/modules/5.18.5-smp/kernel/net/8021q/8021q.ko
INSTALL /lib/modules/5.18.5-smp/kernel/net/ipv6/ipv6.ko
INSTALL /lib/modules/5.18.5-smp/kernel/net/llc/llc.ko
INSTALL /lib/modules/5.18.5-smp/kernel/net/rfkill/rfkill.ko
INSTALL /lib/modules/5.18.5-smp/kernel/net/wireless/cfg80211.ko
INSTALL /lib/modules/5.18.5-smp/kernel/sound/ac97_bus.ko
INSTALL /lib/modules/5.18.5-smp/kernel/sound/core/snd-pcm.ko
INSTALL /lib/modules/5.18.5-smp/kernel/sound/core/snd-timer.ko
INSTALL /lib/modules/5.18.5-smp/kernel/sound/core/snd.ko
INSTALL /lib/modules/5.18.5-smp/kernel/sound/pci/ac97/snd-ac97-codec.ko
INSTALL /lib/modules/5.18.5-smp/kernel/sound/pci/snd-intel8x0.ko
INSTALL /lib/modules/5.18.5-smp/kernel/sound/soundcore.ko
DEPMOD  /lib/modules/5.18.5-smp
root@slack:/usr/src/linux-5.18.5#

```

Rysunek 29 Wynik instalacji modułów

Po pomyślnej instalacji przekopiuję pliki do folderu `/boot`. Następnie do niego przechodzę i podmieniam tablicę na nową.

```

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 30 Kopiowanie plików do /boot i usunięcie System.map

Znowu wygeneruję komendę do utworzenia dysku RAM.

```
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
```

Rysunek 31 Wygenerowanie polecenia

Edytuję wygenerowane polecenie tak aby znowu nie nadpisać pliku 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
49039 bloków
/boot/initrd-new-5.18.5-smp.gz created.
Be sure to run lilo again if you use it.
root@slack:/boot#
```

Rysunek 32 Wynik wygenerowanej komendy

Tak jak poprzednio otwieram plik konfiguracyjny lilo za pomocą nano. Należy w tym pliku wstawić wpis między sekcjami Linux bootable.

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

Rysunek 33 /etc/lilo.conf po edycji

Następnie uruchamiam lilo.

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

Rysunek 34 Wynik polecenia lilo

Wyskoczył jakiś warning, ale w niczym nie przeszkadza i ignoruję. Należy teraz sprawdzić czy wszystko działa. Restartuję w tym celu maszynę.



Rysunek 35 Sukces nowej metody!

Jak widać mogę już wybrać opcję kernel-new. Po wybraniu nowego jądra wyświetliły się pocieszne pingwinki, gdy wszystko się załadowało postanowiłem się zalogować i wywołać komendę neofetch.

