

Kompilacja jądra Linux

Dmytro Nimchynskyi

Najpierw pobieram najnowsze źródła jądra Linux dla gałęzi stabilnej ze strony <https://www.kernel.org>.

W momencie pisania raportu najnowsza wersja to **5.18.4**.



Zmieniam dyrektywę na **/usr/src** i pobieram plik źródłowy polecienniem **wget**
<https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.4.tar.xz>

```
root@localhost:~# cd /usr/src
root@localhost:/usr/src# wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.4.tar.xz
--2022-06-15 23:22:15--  https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.4.tar.xz
Translacja cdn.kernel.org (cdn.kernel.org)... 151.101.113.176, 2a04:4e42:1b::432
Łączenie się z cdn.kernel.org (cdn.kernel.org)|151.101.113.176|:443... połączono.
Zadanie HTTP wyštano, oczekiwanie na odpowiedź... 200 OK
Długość: 129853184 (124M) [application/x-xz]
Zapis do: `linux-5.18.4.tar.xz'

linux-5.18.4.tar.xz          100%[=====] 123,8
4M  199KB/s   w 11m 41s

2022-06-15 23:33:56 (181 KB/s) - zapisano `linux-5.18.4.tar.xz' [129853184/129853184]

root@localhost:/usr/src# ls
linux@ linux-5.15.38/  linux-5.17.9/  linux-5.17.9.tar.xz  linux-5.18.4.tar.xz
root@localhost:/usr/src# |
```

Rozpakowuję plik z źródłami jądra polecienniem **tar -xvpf linux-5.18.4.tar.xz**
Proces rozpakowania zajął mi ok. 1 minutę. Po rozpakowaniu dostałem folder **linux-5.18.4** z taką zawartością.

```
root@localhost:/usr/src# ls
linux@ linux-5.15.38/ linux-5.17.9/ linux-5.17.9.tar.xz linux-5.18.4/ linux-5.18.4.tar.xz
root@localhost:/usr/src# cd linux-5.18.4
root@localhost:/usr/src/linux-5.18.4# ls
COPYING      Kbuild      MAINTAINERS  arch/   crypto/   include/   kernel/   net/     security/  usr/
CREDITS      Kconfig     Makefile    block/   drivers/  init/     lib/     samples/   sound/    virt/
Documentation/ LICENSES/ README     certs/   fs/       ipc/     mm/     scripts/  tools/
root@localhost:/usr/src/linux-5.18.4# |
```

Teraz skopiuję obecną konfigurację jądra i użyję jej jako bazy do zbudowania nowego za pomocą polecenia **zcat /proc/config.gz > .config**. Mam teraz obecną konfigurację jądra w pliku **.config**.

```
root@localhost:/usr/src/linux-5.18.4# zcat /proc/config.gz > .config
root@localhost:/usr/src/linux-5.18.4# ls -a
./          .config          .mailmap        Kbuild        Makefile  certs/   include/   lib/     scripts/  usr/
..          .get_maintainer.ignore  COPYING        Kconfig      README   crypto/  init/     mm/     security/ virt/
.clang-format .gitattributes    CREDITS      LICENSES/   arch/   drivers/  ipc/     net/     sound/
.cocciconfig   .gitignore      Documentation/ MAINTAINERS block/   fs/       kernel/  samples/ tools/
root@localhost:/usr/src/linux-5.18.4# |
```

Na wszelki wypadek zrobię backup konfiguracji poleceniem **cp .config config_bak**.

```
root@localhost:/usr/src/linux-5.18.4# cp .config config_bak
root@localhost:/usr/src/linux-5.18.4# ls -a
./          .config      .mailmap      Kbuild      Makefile   certs/    fs/       kernel/   samples/   tools/
..          .get_maintainer.ignore  COPYING      KConfig     README    config_bak  include/   lib/      scripts/   usr/
.clang-format .gitattributes  CREDITS     LICENSES/   arch/    crypto/    init/     mm/      security/  virt/
.cocciconfig  .gitignore     Documentation/ MAINTAINERS  block/   drivers/   ipc/      net/      sound/
root@localhost:/usr/src/linux-5.18.4#
```

Od tego momentu zacznę generować pliki konfiguracyjne – najpierw za pomocą starej metody **localmodconfig** i potem za pomocą nowej metody **streamline_config.pl**.

Przebieg procesu komplikacji dla metody starej

Używając polecenia **make localmodconfig** wygeneruję plik konfiguracyjny na podstawie bieżącej konfiguracji i załadowanych modułów. Wszystkie inne ustawienia zostawię domyślnie.

```
root@localhost:/usr/src/linux-5.18.4# make localmodconfig
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
Old Idle dynticks config (NO_HZ) [Y/n/?] y
High Resolution Timer Support (HIGH_RES_TIMERS) [Y/n/?] y
Clocksource watchdog maximum allowable skew (in µs) (CLOCKSOURCE_WATCHDOG_MAX_SKEW_US) [100] (NEW) |
```

```
ssh root@127.0.0.1 -p 2222
Test functions located in the hexdump module at runtime (TEST_HEXDUMP) [N/m/y/?] n
Test string functions at runtime (TEST_SELFTEST) [N/m/y/?] n
Test functions located in the string_helpers module at runtime (TEST_STRING_HELPERS) [N/m/y/?] n
Test strcpy*() 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 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 (TEST_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@localhost:/usr/src/linux-5.18.4#
```

Ten proces szybko się zakończył sukcesem i wygenerował plik **.config**, starą konfigurację zapisał do **.config.old**.

```
ssh root@127.0.0.1 -p 2222
root@localhost:/usr/src/linux-5.18.4# ls -a
./          .config          .gitignore  Documentation/  MAINTAINERS  block/      drivers/   ipc/      net/      sound/
..          .config.old       .mailmap    Kbuild      Makefile    certs/     fs/       kernel/   samples/  tools/
.clang-format .get_maintainer.ignore  COPYING    Kconfig     README    config_bak  include/  lib/      scripts/  usr/
.cocciconfig  .gitattributes      CREDITS   LICENSES/  arch/     crypto/    init/    mm/      security/ virt/
root@localhost:/usr/src/linux-5.18.4#
```

Poleceniem **lsmod** sprawdzę wszystkie załadowane moduły.

```
root@localhost:/usr/src/linux-5.18.4# lsmod
Module           Size  Used by
videodev        24276   0
drm_vram_helper 29480   1 vboxvideo
drm_ttm_helper  16384   2 drm_vram_helper,vboxvideo
cfg80211       77048   0
8021q          28872   0
gbpf            10354   1 8021q
mfp             20480   1 8021q
stp              16384   1 garp
llc              16384   2 garp,stp
rfkill           24575   1 fgf80211
ipv6             65872   22
intel_rapl_msr 20480   0
joydev           20480   0
intel_rapl_common 24576   1 intel_rapl_msr
vmwgfx          21916   0
ttm              61440   3 drm_vram_helper,drm_ttm_helper,vmwgfx
crc32_pclmul    16384   0
drm_kms_helper  237568  3 drm_vram_helper,vboxvideo,vmwgfx
rmi              30048   0
intel_cstate     20480   0
snd_intel8x0    36864   2
snd_ac97_codec  122880  1 snd_intel8x0
psmouse          12080   0
drm              475126  9 drm_vram_helper,drm_ttm_helper,vboxvideo,vmwgfx,ttm,drm_kms_helper
snd_pcm          102400  2 snd_ac97_codec,snd_intel8x0
snd_timer        32768   1 snd_pcm
evdev            20480   13
snd              73728   8 snd_ac97_codec,snd_timer,snd_intel8x0,snd_pcm
pcnet32          45856   0
fb_sys_fops     16384   1 drm_kms_helper
syscopyarea     16384   1 drm_kms_helper
soundcore        16384   1 snd
i2c              16384   1 i2c_piix4
sysfillrect     16384   1 drm_kms_helper
i2c_piix4        20480   0
serio_raw        16384   0
nouveau          16384   1 snd_ac97_codec
sysmplt          16384   1 drm_kms_helper
ohci_pci         16384   0
vboxguest        32768   6
i2c_core         73728   1 i2c_piix4,psmouse,drm_kms_helper,drm
ohci_hd          30684   1 ohci_pci
ehci_pci         16384   0
intel_agp        16384   0
ehci_hcd         53248   1 ehci_pci
intel_igt        20480   1 intel_agp
appgt            49960   4 intel_agp,intel_gtt,ttm,drm
ac               16384   0
battery          20480   0
vicode           49152   0
button            16384   0
loop              32768   0
root@localhost:/usr/src/linux-5.18.4#
```

Teraz można kompliować obraz jądra. Dla tego użyję polecenie **make bzImage** z parametrem **-j4** żeby wykorzystać 4 jądra mojego systemu dla procesu komplikacji.

```
root@localhost:/usr/src/linux-5.18.4# make -j4 bzImage
SYNC  include/config/auto.conf.cmd
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/ioctl.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
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/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
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
WRAP  arch/x86/include/generated/asm/mmiohw.h
WRAP  arch/x86/include/generated/asm/module.lds.h
WRAP  arch/x86/include/generated/asm/unaligned.h
WRAP  arch/x86/include/generated/asm/rwonce.h
UPD   include/generated/uapi/linux/version.h
```

Proces komplikacji zajął 27 min. Podczas komplikacji mój laptop stał się zauważalnie cieplejszy (~79°C), a wentylatory zaczęły się kręcić(7200rpm).

```
● ● ◉ └─SSH1
root@localhost:~# make bzImage
arch/x86/boot/compressed/vmlinux.lds
AS arch/x86/boot/pnjmp.o
AS arch/x86/boot/compressed/kernel_info.o
CC arch/x86/boot/printf.o
AS arch/x86/boot/compressed/head_32.o
VOFFSET arch/x86/boot/compressed../voffset.h
CC arch/x86/boot/regs.o
CC arch/x86/boot/compressed/string.o
CC arch/x86/boot/tty.o
CC arch/x86/boot/compressed/cmdline.o
CC arch/x86/boot/compressed/error.o
CC arch/x86/boot/video.o
OBJCOPY arch/x86/boot/compressed/vmlinux.bin
RELOCS arch/x86/boot/compressed/vmlinux.relocs
CC arch/x86/boot/video-mode.o
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/version.o
CC arch/x86/boot/compressed/acpi.o
CC arch/x86/boot/video-vga.o
CC arch/x86/boot/video-vesa.o
CC arch/x86/boot/video-macos.o
HOSTCC arch/x86/boot/tools/build
CC arch/x86/boot/compressed/misc.o
CPUSTR arch/x86/boot/cpustr.h
CC arch/x86/boot/cpu.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@localhost:/usr/src/linux-5.18.4#
```

Na szczęście komplikacja zakończyła się sukcesem. Teraz przechodzę do zbudowania modułów za pomocą polecenia **make modules**.

```
● ● ◉ └─SSH1
root@localhost:/usr/src/linux-5.18.4# make modules
CALL scripts/checksyscalls.sh
CALL scripts/atomic/check-atomics.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
AS [M] arch/x86/crypto/crc32-pclmul_asn.o
CC [M] arch/x86/crypto/crc32-pclmul_glue.o
LD [M] arch/x86/crypto/crc32-pclmul.o
CC [M] drivers/video/fbdev/core/sysfillrect.o
CC [M] drivers/video/fbdev/core/syscopyarea.o
CC [M] drivers/video/fbdev/core/sysimblt.o
CC [M] drivers/video/fbdev/core/fb_sys_fops.o
CC [M] drivers/acpi/ac.o
CC [M] drivers/acpi/button.o
CC [M] drivers/acpi/acpi_video.o
CC [M] drivers/acpi/video_detect.o
LD [M] drivers/acpi/video.o
CC [M] drivers/acpi/battery.o
CC [M] drivers/char/agp/backend.o
CC [M] drivers/char/agp/generic.o
CC [M] drivers/char/agp/isoch.o
CC [M] drivers/char/agp/frontend.o
LD [M] drivers/char/agp/aggpart.o
CC [M] drivers/char/agp/intel-agp.o
CC [M] drivers/char/agp/intel-gtt.o
CC [M] drivers/gpu/drm/ttm/ttm_tt.o
CC [M] drivers/gpu/drm/ttm/tm_bo.o
CC [M] drivers/gpu/drm/ttm/tm_bo_util.o
CC [M] drivers/gpu/drm/ttm/tm_bo_vm.o
CC [M] drivers/gpu/drm/ttm/tm_module.o
CC [M] drivers/gpu/drm/ttm/tm_execbuf_util.o
CC [M] drivers/gpu/drm/ttm/tm_range_manager.o
CC [M] drivers/gpu/drm/ttm/tm_resource.o
CC [M] drivers/gpu/drm/ttm/tm_pool.o
CC [M] drivers/gpu/drm/ttm/tm_device.o
CC [M] drivers/gpu/drm/ttm/tm_sys_manager.o
CC [M] drivers/gpu/drm/ttm/tm_agp_backend.o
LD [M] drivers/gpu/drm/ttm/ttm.o
CC [M] drivers/gpu/drm/vmwgfx/vmwgfx_execbuf.o
CC [M] drivers/gpu/drm/vmwgfx/vmwgfx_gmr.o
```

Podczas budowania modułów, podobnie jak z komplikacją jądra, laptop stał cieplejszy (~74°C), a wentylatory zaczęły się kręcić (5500rpm).

```
 ③ 2361 ssh root@127.0.0.1 -p 2222
CC [M] drivers/video/fbdev/core/sysfillrect.mod.o
LD [M] drivers/video/fbdev/core/sysfillrect.ko
CC [M] drivers/video/fbdev/core/sysimgblt.mod.o
LD [M] drivers/video/fbdev/core/sysimgblt.ko
CC [M] drivers/virt/vboxguest/vboxguest.mod.o
LD [M] drivers/virt/vboxguest/vboxguest.ko
CC [M] net/802/garp.mod.o
LD [M] net/802/garp.ko
CC [M] net/802/mrp.mod.o
LD [M] net/802/mrp.ko
CC [M] net/802/p8022.mod.o
LD [M] net/802/p8022.ko
CC [M] net/802/psnap.mod.o
LD [M] net/802/psnap.ko
CC [M] net/802/stp.mod.o
LD [M] net/802/stp.ko
CC [M] net/8021q/8021q.mod.o
LD [M] net/8021q/8021q.ko
CC [M] net/ipv6/ipv6.mod.o
LD [M] net/ipv6/ipv6.ko
CC [M] net/lld/lld.mod.o
LD [M] net/lld/lld.ko
CC [M] net/rfkill/rfkill.mod.o
LD [M] net/rfkill/rfkill.ko
CC [M] net/wireless/cfg80211.mod.o
LD [M] net/wireless/cfg80211.ko
CC [M] sound/ac97_bus.mod.o
LD [M] sound/ac97_bus.ko
CC [M] sound/core/snd-pcm.mod.o
LD [M] sound/core/snd-pcm.ko
CC [M] sound/core/snd-timer.mod.o
LD [M] sound/core/snd-timer.ko
CC [M] sound/core/snd.mod.o
LD [M] sound/core/snd.ko
CC [M] sound/pci/ac97/snd-ac97-codec.mod.o
LD [M] sound/pci/ac97/snd-ac97-codec.ko
CC [M] sound/pci/snd-intel8x0.mod.o
LD [M] sound/pci/snd-intel8x0.ko
CC [M] sound/soundcore.mod.o
LD [M] sound/soundcore.ko
root@localhost:/usr/src/linux-5.18.4# |
```

Proces budowania modułów też zakończył się sukcesem, trwał on 16 min. Teraz możemy ich zainstalować za pomocą polecenia **make modules_install**.

```
 ④ 2361 ssh root@127.0.0.1 -p 2222
-rw-r--r-- 1 root root 642840 cie 16 02:04 vmlinuz.savers
root@localhost:/usr/src/linux-5.18.4# clear
root@localhost:/usr/src/linux-5.18.4# make modules_install
INSTALL /lib/modules/5.18.4-smp/kernel/arch/x86/events/intel/intel-cstate.ko
INSTALL /lib/modules/5.18.4-smp/kernel/arch/x86/events/rapl.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/api/cie.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/api/drm/i915.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/api/button.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/api/video.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/block/loop.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/char/agp/agpari.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/char/agp/intel_agp.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/char/agp/intel_gt.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/gpu/drm/rm.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/gpu/drm/drm_kms_helper.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/gpu/drm/drm_vam_helper.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/gpu/drm/ttm_ttm.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/gpu/drm/vboxvideo.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/gpu/drm/vt/ttm_vt.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/12c/alpha/12c_alpha.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/12c/busses/12c_p1x4.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/12c/12c-core.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/input/evdev.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/input/hid.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/input/mouse/psmouse.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/input/serio/serio_raw.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/net/ethernet/amd/pcnet32.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/net/ethernet/intel/igb.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/powercap/intel_rapl_common.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/powercap/intel_rapl_lsnr.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/usb/host/ehci-hcd.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/usb/host/ohci-hcd.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/usb/host/ohci-pci.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/video/fbdev/core/fb_sys_fops.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/video/fbdev/core/sysfillrect.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/video/fbdev/core/sysimgblt.ko
INSTALL /lib/modules/5.18.4-smp/kernel/drivers/virt/vboxguest/vboxguest.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/802/garp.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/802/mrp.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/802/p8022.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/802/psnap.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/802/stp.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/8021q/8021q.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/ipv6/ipv6.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/lld/lld.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/rfkill/rfkill.ko
INSTALL /lib/modules/5.18.4-smp/kernel/net/wireless/cfg80211.ko
INSTALL /lib/modules/5.18.4-smp/kernel/sound/core/snd-pcm.ko
INSTALL /lib/modules/5.18.4-smp/kernel/sound/core/snd-timer.ko
INSTALL /lib/modules/5.18.4-smp/kernel/sound/core/snd.ko
INSTALL /lib/modules/5.18.4-smp/kernel/sound/core/snd-ac97-codec.ko
INSTALL /lib/modules/5.18.4-smp/kernel/sound/pci/snd-intel8x0.ko
INSTALL /lib/modules/5.18.4-smp/kernel/sound/soundcore.ko
DEPMOD /lib/modules/5.18.4-smp
root@localhost:/usr/src/linux-5.18.4# |
```

Instalacja modułów zakończyła się bardzo szybko. Wyżej jest wynik polecenia.

Teraz wpiszę polecenie **ls /lib/modules/5.18.4-smp** żeby sprawdzić listę zainstalowanych modułów.

```
root@localhost:/usr/src/linux-5.18.4# ls /lib/modules/5.18.4-smp
build@           modules.alias.bin      modules.builtin.bin    modules.dep.bin   modules.softdep      source@
kernel/         modules.builtin      modules.builtin.modinfo  modules.devname  modules.symbols
modules.alias   modules.builtin.alias.bin  modules.dep        modules.order    modules.symbols.bin
root@localhost:/usr/src/linux-5.18.4#
```

Dalej przekopuję następujące pliki jądra do katalogu **/boot**:

- Obraz jądra
cp arch/x86/boot/bzImage /boot/vmlinuz-custom-5.18.4-smp
- Tablica symboli używana przez kernel
cp System.map /boot/System.map-custom-5.18.4-smp
- Plik konfiguracyjny kernela
cp .config /boot/config-custom-5.18.4-smp

```
root@localhost:/usr/src/linux-5.18.4# cp arch/x86/boot/bzImage /boot/vmlinuz-custom-5.18.4-smp
root@localhost:/usr/src/linux-5.18.4# cp System.map /boot/System.map-custom-5.18.4-smp
root@localhost:/usr/src/linux-5.18.4# cp .config /boot/config-custom-5.18.4-smp
root@localhost:/usr/src/linux-5.18.4#
```

Utworzę link symboliczny w systemie dla tablicy symboli kernela z usunięciem starej.

```
root@localhost:/usr/src/linux-5.18.4# cd /boot
root@localhost:/boot# rm System.map
root@localhost:/boot# ln -s System.map-custom-5.18.4-smp System.map
root@localhost:/boot#
```

Następnie utworzę dysk RAM.

```
root@localhost:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.4-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 5.18.4-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
root@localhost:/boot# mkinitrd -c -k 5.18.4-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
49031 blokow
/boot/initrd.gz created.
Be sure to run lilo again if you use it.
root@localhost:/boot#
```

Jak widać skrypt **/usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.4-smp** wygenerował polecenie **mkinitrd -c -k 5.18.4-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz** za pomocą jakiego udało mi się stworzyć dysk RAM.

Teraz dodaję do konfiguracji bootloadera lilo nowy wpis

```
GNU nano 6.0                               ssh root@127.0.0.1 -p 2222
/etc/lilo.conf

#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/sda1
label = "Slackware 15.0"
read-only

image = /boot/vmlinuz-custom-5.17.9-smp
root = /dev/sda1
initrd = /boot/initrd-custom-5.17.9-smp.gz
label = "kernel-custom"
read-only

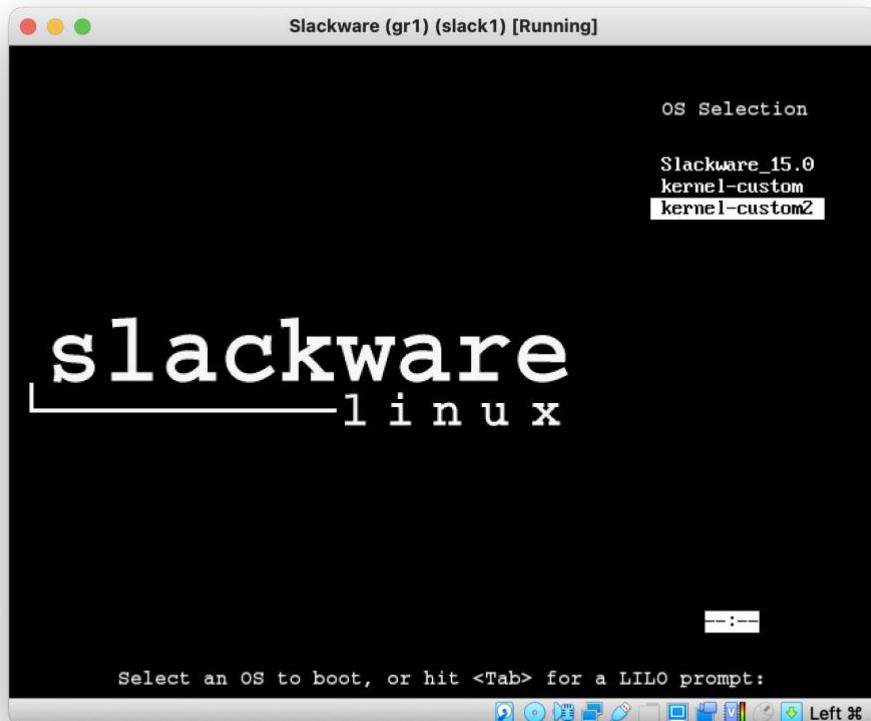
image = /boot/vmlinuz-custom-5.18.4-smp
root = /dev/sda1
initrd = /boot/initrd.gz
label = "kernel-custom2"
read-only
# Linux bootable partition config ends
| [ Zapisano 81 linii ]
^G Pomoc      ^O Zapisz    ^W Wyszukaj   ^K Wytnij    ^T Wykonaj   ^C Lokalizacja M-U Odwołaj   M-A Ustaw znacz M-] Do nawiasu
^X Wyjdź      ^R Wczyt.plik ^N Zastąp     ^U Wklej     ^J Wyjustuj  ^Y Do linii  M-E Odtwórz   M-G Kopiuj   ^Q Gdzie było
```

Zapisuję zmiany polecienniem **lilo**.

```
root@localhost:/boot# lilo
Warning: LBA32 addressing assumed
Added Slackware_15.0 *
Added kernel-custom +
Added kernel-custom2 +
One warning was issued.
root@localhost:/boot# | [ Zapisano 81 linii ]
```

I już mogę zrestartować system.

Jak widać na screenach przy uruchomieniu mam opcję **kernel-custom2** jaką ja poprzednio dodałem do konfiguracji bootloadera.



Wybieram ją. System uruchamie się poprawnie.

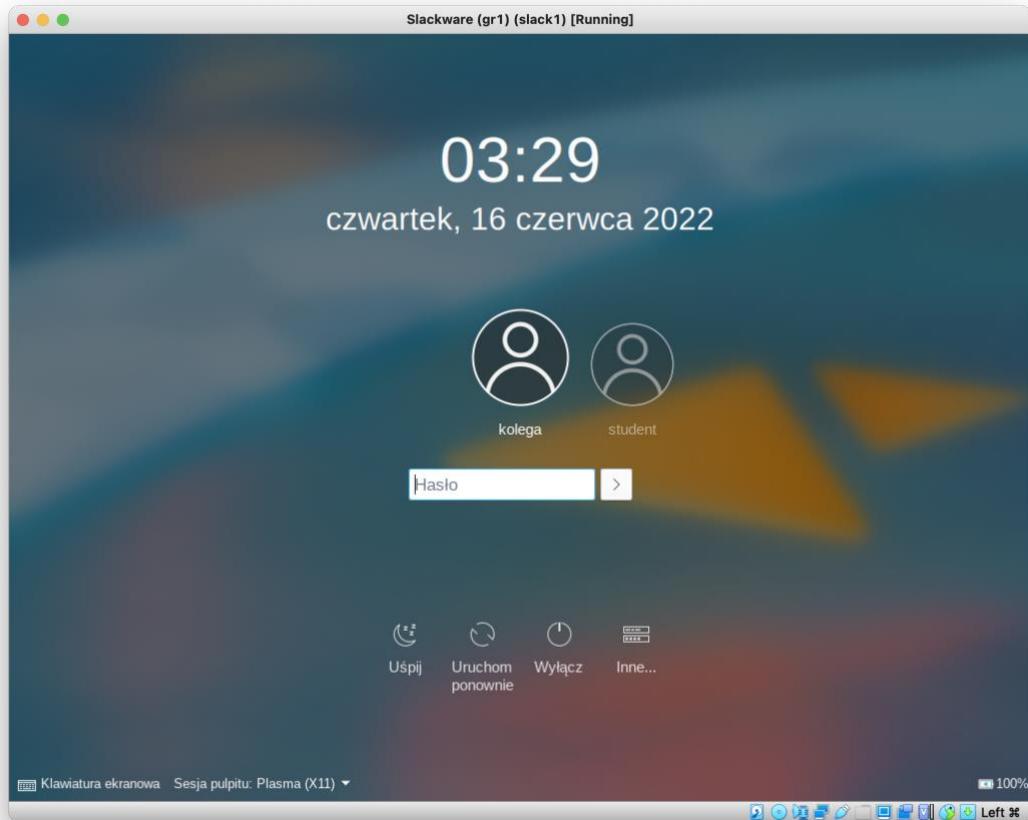
```
Slackware (gr1) (slack1) [Running]
VirtualBox Guest Additions: Building the modules for kernel 5.15.38-smp.
The Virtual Machine reports that the guest OS does not support mouse pointer integration in the current video mode. You need to capture the mouse (by clicking over the VM).
VirtualBox Guest Additions: Look at /var/log/vboxadd-setup.log to find out what went wrong.
VirtualBox Guest Additions: Running kernel modules will not be replaced until the system is restarted.
vboxadd-service.sh: Starting VirtualBox Guest Addition service.
Starting up X11 session manager...

Welcome to Linux 5.15.38-smp i686 (tty1)

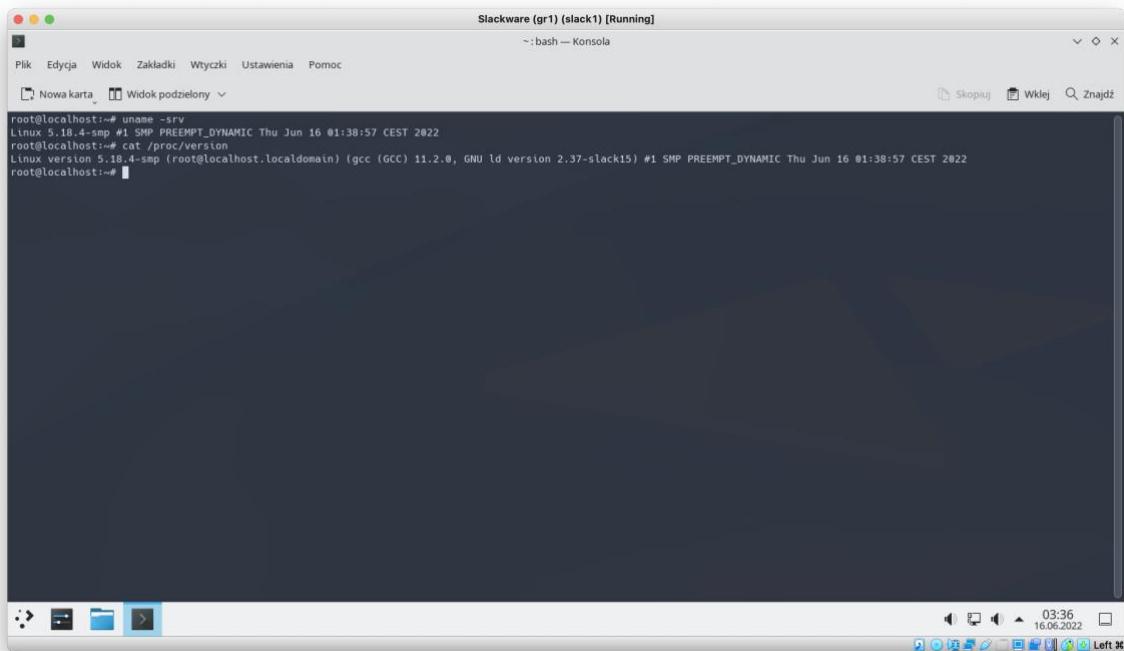
localhost login: vboxadd-service.sh: Stopping VirtualBox Guest Addition service.
vboxadd-service.sh: VirtualBox Guest Addition service failed to stop.
VirtualBox Guest Additions: Stopping.
VirtualBox Guest Additions: Building the modules for kernel 5.18.4-smp.

VirtualBox Guest Additions: Look at /var/log/vboxadd-setup.log to find out what went wrong.
VirtualBox Guest Additions: You may need to restart your guest system to finish removing guest drivers.
Running shutdown script /etc/rc.d/rc.6:
INIT: version 3.01 reloading
Saving system time to the hardware clock (localtime).
Stopping havedged.
Unmounting remote filesystems:
Stopping system message bus...
Stopping the network interfaces...
Stopping udevd
Sending all processes the SIGTERM signal.
Sending all processes the SIGKILL signal.
Saving random seed from /dev/urandom in /etc/random-seed.
Turning off swap.
Unmounting local file systems:
/slax : successfully unmounted
/home : successfully unmounted
/dev/shm : ignored
/dev/pts : successfully unmounted
/run : ignored
/sys : ignored
/proc : ignored
/dev : ignored
[21090.023023] EXT4-fs (sda1): re-mounted. Opts: (null). Quota mode: none.
Remounting root filesystem read-only:
[21090.047483] EXT4-fs (sda1): re-mounted. Opts: (null). Quota mode: none.
mount: /dev/sda1 mounted on /
```

Loguję się do systemu.



Poleceniami **uname -srv i cat /proc/version** sprawdzam wersję jądra.



Przebieg procesu komplikacji dla metody nowej

Rozpakuję plik z źródłami jądra jeszcze raz ale już z nową nazwą **linux-5-18-4-new**.

```
root@localhost:/usr/src# mkdir linux-5.18.4-new
root@localhost:/usr/src# ls
linux@    linux-5.15.38.zip  linux-5.17.9.tar.xz  linux-5.18.4-new/  linux-5.18.4.tar.xz
linux-5.15.38/ linux-5.17.9/  linux-5.18.4/      linux-5.18.4.tar.gz
root@localhost:/usr/src# tar -xf linux-5.18.4.tar.gz --strip-components=1 -C linux-5.18.4-new/
root@localhost:/usr/src# cd linux-5.18.4-new/
root@localhost:/usr/src/linux-5.18.4-new# ls -a
./               .missing-syccals.d   .vmlinux.cmd   System.map   kernel/           security/
..               .modules-only.symvers.cmd COPYING       arch/        lib/             sound/
.Module.symvers.cmd .modules.order.cmd CREDITS     block/      linux-5.18.4-compressed tools/
.clang-format     .tmp_System.map     Documentation/ certs/      mm/
.cocciconfig      .tmp_vmlinux.kallsyms1* Kbuild      config_bak  modules-only.symvers virt/
.config          .tmp_vmlinux.kallsyms1.S Kconfig      crypto/    modules.builtin  vmlinux*
.config.old       .tmp_vmlinux.kallsyms1.o LICENSES/   drivers/   modules.builtin.modinfo vmlinux.o
.get_maintainer.ignore .tmp_vmlinux.kallsyms2* MAINTAINERS  fs/       modules.order   vmlinux.symvers
.gitattributes    .tmp_vmlinux.kallsyms2.S Makefile     include/  net/
.gitignore        .tmp_vmlinux.kallsyms2.o Module.symvers init/    samples/
.mailmap          .version            README     ipc/      scripts/
root@localhost:/usr/src/linux-5.18.4-new# |
```

Teraz skopiuję obecną konfigurację jądra za pomocą polecenia **zcat /proc/config.gz > .config** i używając polecenia **./scripts/kconfig/streamline_config.pl > config_strip** wygeneruję plik konfiguracyjny z wyłączeniem wszystkich modułów, które nie są załadowane do mojego systemu.

```
root@localhost:/usr/src/linux-5.18.4-new# zcat /proc/config.gz > .config
root@localhost:/usr/src/linux-5.18.4-new# ls -a
./               .missing-syccals.d   .vmlinux.cmd   System.map   kernel/           security/
..               .modules-only.symvers.cmd COPYING       arch/        lib/             sound/
.Module.symvers.cmd .modules.order.cmd CREDITS     block/      linux-5.18.4-compressed tools/
.clang-format     .tmp_System.map     Documentation/ certs/      mm/
.cocciconfig      .tmp_vmlinux.kallsyms1* Kbuild      config_bak  modules-only.symvers virt/
.config          .tmp_vmlinux.kallsyms1.S Kconfig      crypto/    modules.builtin  vmlinux*
.config.old       .tmp_vmlinux.kallsyms1.o LICENSES/   drivers/   modules.builtin.modinfo vmlinux.o
.get_maintainer.ignore .tmp_vmlinux.kallsyms2* MAINTAINERS  fs/       modules.order   vmlinux.symvers
.gitattributes    .tmp_vmlinux.kallsyms2.S Makefile     include/  net/
.gitignore        .tmp_vmlinux.kallsyms2.o Module.symvers init/    samples/
.mailmap          .version            README     ipc/      scripts/
root@localhost:/usr/src/linux-5.18.4-new# ./scripts/kconfig/streamline_config.pl > config_strip
using config: ".config"
root@localhost:/usr/src/linux-5.18.4-new# ls -a
./               .missing-syccals.d   .vmlinux.cmd   System.map   ipc/           scripts/
..               .modules-only.symvers.cmd COPYING       arch/        kernel/         security/
.Module.symvers.cmd .modules.order.cmd CREDITS     block/      lib/           sound/
.clang-format     .tmp_System.map     Documentation/ certs/      linux-5.18.4-compressed tools/
.cocciconfig      .tmp_vmlinux.kallsyms1* Kbuild      config_bak  mm/
.config          .tmp_vmlinux.kallsyms1.S Kconfig      config_strip  modules-only.symvers virt/
.config.old       .tmp_vmlinux.kallsyms1.o LICENSES/   drivers/   modules.builtin  vmlinux*
.get_maintainer.ignore .tmp_vmlinux.kallsyms2* MAINTAINERS  fs/       modules.builtin.modinfo vmlinux.o
.gitattributes    .tmp_vmlinux.kallsyms2.S Makefile     include/  net/
.gitignore        .tmp_vmlinux.kallsyms2.o Module.symvers init/    samples/
.mailmap          .version            README     init/      vmlinux.symvers
root@localhost:/usr/src/linux-5.18.4-new# |
```

Proces szybko się zakończył sukcesem i wygenerował plik z konfiguracją **config_strip**.

Robię backup starej konfiguracji i przenoszę konfigurację **config_strip** do **.config**.

```
root@localhost:/usr/src/linux-5.18.4-new# cp .config config_bak
root@localhost:/usr/src/linux-5.18.4-new# mv config_strip .config
root@localhost:/usr/src/linux-5.18.4-new# ls -a
./               .missing-sysscls.d   .vmlinux.cmd    System.map   kernel/           security/
..               .modules-only.symvers.cmd COPYING      arch/       lib/            sound/
.Module.symvers.cmd .modules.order.cmd CREDITS     block/     linux-5.18.4-compressed tools/
.clang-format     .tmp_System.map    Documentation/ certs/     mm/
.cocciconfig      .tmp_vmlinux.kallsyms1* Kbuild     config_bak  modules-only.symvers virt/
.config          .tmp_vmlinux.kallsyms1.S Kconfig    crypto/    modules.builtin  vmlinux*
.config.old       .tmp_vmlinux.kallsyms1.o LICENSES/   drivers/   modules.builtin.modinfo vmlinux.o
.get_maintainer.ignore .tmp_vmlinux.kallsyms2* MAINTAINERS  fs/        modules.order   vmlinux.symvers
.gitattributes    .tmp_vmlinux.kallsyms2.S Makefile   include/  net/
.gitignore         .tmp_vmlinux.kallsyms2.o Module.symvers init/    samples/
.mailmap          .version          README     ipc/      scripts/
root@localhost:/usr/src/linux-5.18.4-new# |
```

Teraz mogę zaczynać budowanie pliku konfiguracyjnego jądra za pomocą polecenia **make oldconfig**.

```
root@localhost:/usr/src/linux-5.18.4-new# make oldconfig
*
* 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
Old Idle dynticks config (NO_HZ) [Y/n/?] y
High Resolution Timer Support (HIGH_RES_TIMERS) [Y/n/?] y
Clocksource watchdog maximum allowable skew (in µs) (CLOCKSOURCE_WATCHDOG_MAX_SKEW_US) [100] (NEW)
```

Wszystkie ustawienia pozostawiam domyślnie.

```

● ● ● ˇˇˇ1
ssh root@127.0.0.1 -p 2222
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_UID) [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_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@localhost:/usr/src/linux-5.18.4-new#

```

Proces zakończył się sukcesem. Konfiguracja została zapisana. Już można zaczynać kompilować obraz jądra. Dla metody nowej proces będzie taki sam jak dla metody starej. Dla tego użyję polecenie **make bzImage** z parametrem **-j4** żeby wykorzystać 4 jądra mojego systemu dla procesu komplikacji.

```

● ● ● ˇˇˇ1
ssh root@127.0.0.1 -p 2222
root@localhost:/usr/src/linux-5.18.4-new# 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@localhost:/usr/src/linux-5.18.4-new#

```

Proces komplikacji zajął 2 min. i zakończył się sukcesem.

Teraz przechodzę do zbudowania modułów za pomocą polecenia `make modules`.

```
root@localhost:/usr/src/linux-5.18.4-new# make modules  
  CALL  scripts/checksyscalls.sh  
  CALL  scripts/atomic/check-atomics.sh  
root@localhost:/usr/src/linux-5.18.4-new# |
```

Proces budowania modułów też się zakończył się sukcesem, trwał on 1 min. Teraz możemy ich zainstalować za pomocą polecenia `make modules_install`.

Instalacja modułów zakończyła się bardzo szybko. Wyżej jest wynik polecenia.

Teraz wpiszę polecenie **ls /lib/modules/5.18.4-smp** żeby sprawdzić listę zainstalowanych modułów.

```
root@localhost:/usr/src/linux-5.18.4-new# ls /lib/modules/5.18.4-smp
build@           modules.alias.bin      modules.builtin.bin   modules.dep.bin  modules.softdep      source@
kernel/          modules.builtin      modules.builtin.modinfo  modules.devname  modules.symbols
modules.alias    modules.builtin.alias.bin  modules.dep        modules.order    modules.symbols.bin
root@localhost:/usr/src/linux-5.18.4-new# |
```

Dalej przekopię następujące pliki jądra do katalogu **/boot**:

- Obraz jądra
cp arch/x86/boot/bzImage /boot/vmlinuz-custom-5.18.4-smp
- Tablica symboli używana przez kernel
cp System.map /boot/System.map-custom-5.18.4-smp
- Plik konfiguracyjny kernela
cp .config /boot/config-custom-5.18.4-smp

```
root@localhost:/usr/src/linux-5.18.4-new# cp arch/x86/boot/bzImage /boot/vmlinuz-custom-5.18.4-smp
root@localhost:/usr/src/linux-5.18.4-new# cp System.map /boot/System.map-custom-5.18.4-smp
root@localhost:/usr/src/linux-5.18.4-new# cp .config /boot/config-custom-5.18.4-smp
root@localhost:/usr/src/linux-5.18.4-new# |
```

Utworzę link symboliczny w systemie dla tablicy symboli kernela z usunięciem starej.

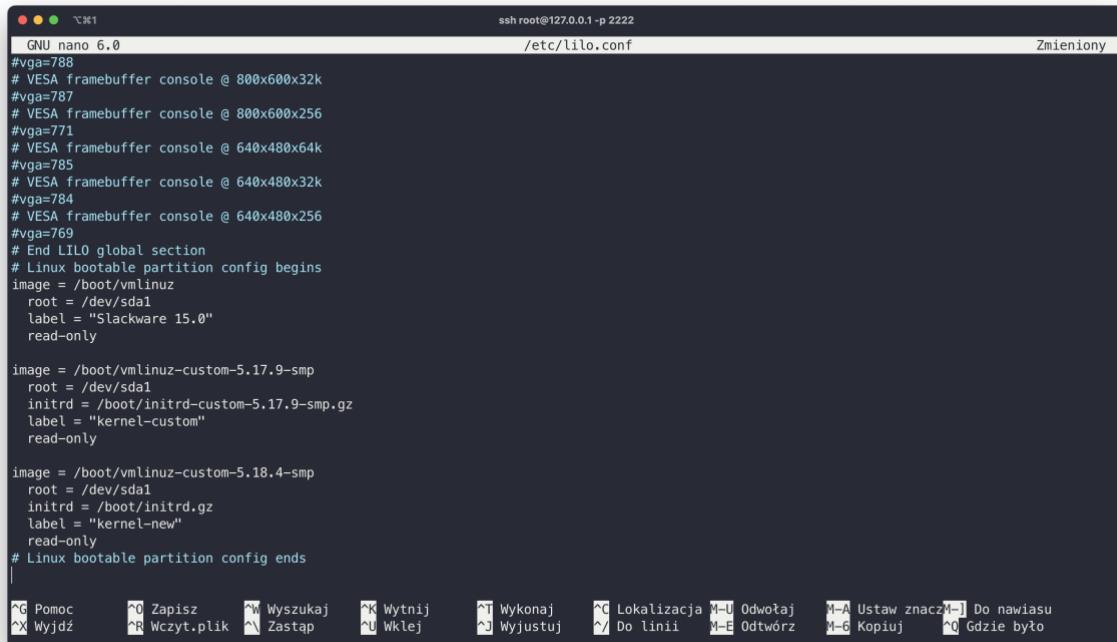
```
ssh root@127.0.0.1 -p 2222
root@localhost:/usr/src/linux-5.18.4-new# cd /boot
root@localhost:/boot# rm System.map
root@localhost:/boot# ln -s System.map-custom-5.18.4-smp System.map
root@localhost:/boot# |
```

Następnie utworzę dysk RAM.

```
ssh root@127.0.0.1 -p 2222
root@localhost:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.4-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 5.18.4-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
root@localhost:/boot# mkinitrd -c -k 5.18.4-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
49031 blokow
/boot/initrd.gz created.
Be sure to run lilo again if you use it.
root@localhost:/boot# |
```

Jak widać skrypt **/usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.4-smp** wygenerował polecenie **mkinitrd -c -k 5.18.4-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz** za pomocą jakiego udało mi się stworzyć dysk RAM.

Teraz dodaję do konfiguracji bootloadera lilo nowy wpis. Dam nazwę **kernel-new**.

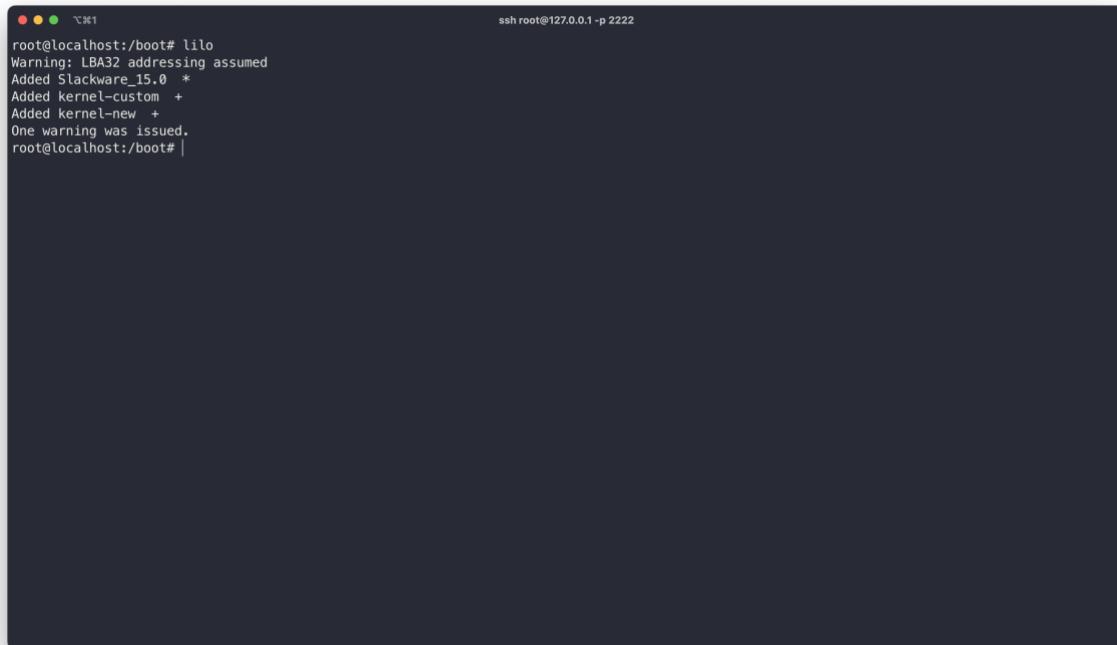


```
GNU nano 6.0                               ssh root@127.0.0.1 -p 2222
/etc/lilo.conf                                Zmieniony
#vga=788
# VESA framebuffer console @ 800x600x32k
#vga=787
# VESA framebuffer console @ 800x600x256
#vga=771
# VESA framebuffer console @ 640x480x64k
#vga=785
# VESA framebuffer console @ 640x480x32k
#vga=784
# VESA framebuffer console @ 640x480x256
#vga=769
# End LILO global section
# Linux bootable partition config begins
image = /boot/vmlinuz
root = /dev/sda1
label = "Slackware 15.0"
read-only

image = /boot/vmlinuz-custom-5.17.9-smp
root = /dev/sda1
initrd = /boot/initrd-custom-5.17.9-smp.gz
label = "kernel-custom"
read-only
# Linux bootable partition config ends
|
```

Pomoc Zapisz Wyszukaj Wytnij Wykonaj Lokalizacja Odwołaj Ustaw znacz Do nawiasu
Wyjdź Wczyt.plik Zastąp Wklej Wyjustuj Do linii Odtwórz Kopij Gdzie było

Zapisuję zmiany polecienniem **lilo**.



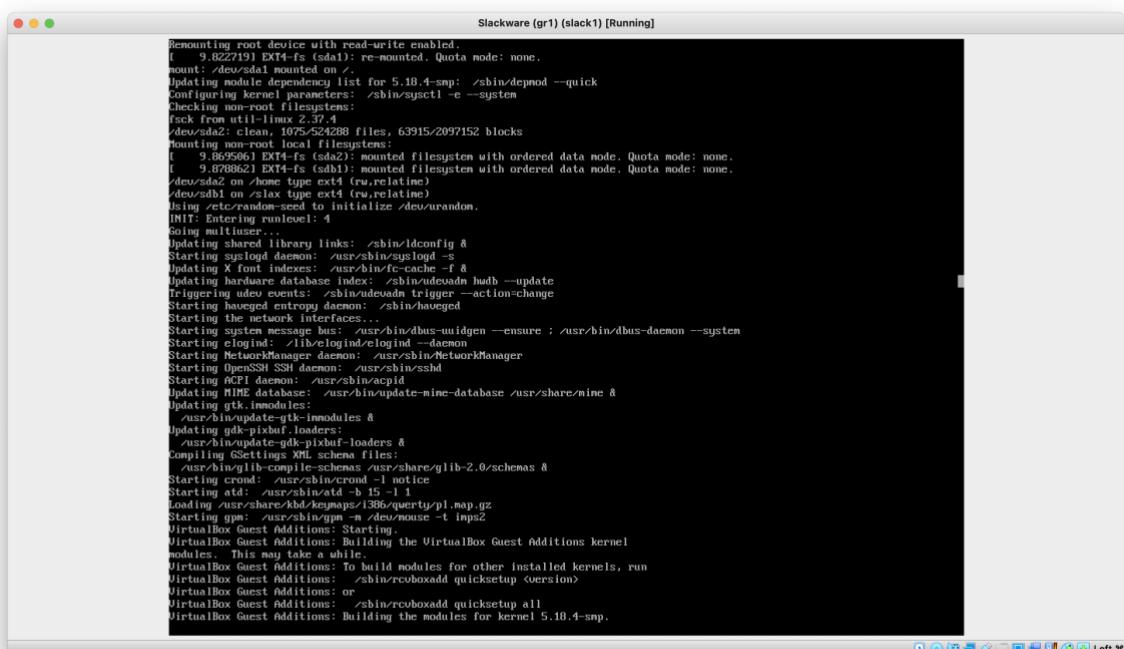
```
root@localhost:/boot# lilo
Warning: LBA32 addressing assumed
Added Slackware_15.0 *
Added kernel-custom +
One warning was issued.
root@localhost:/boot# |
```

I już mogę zrestartować system.

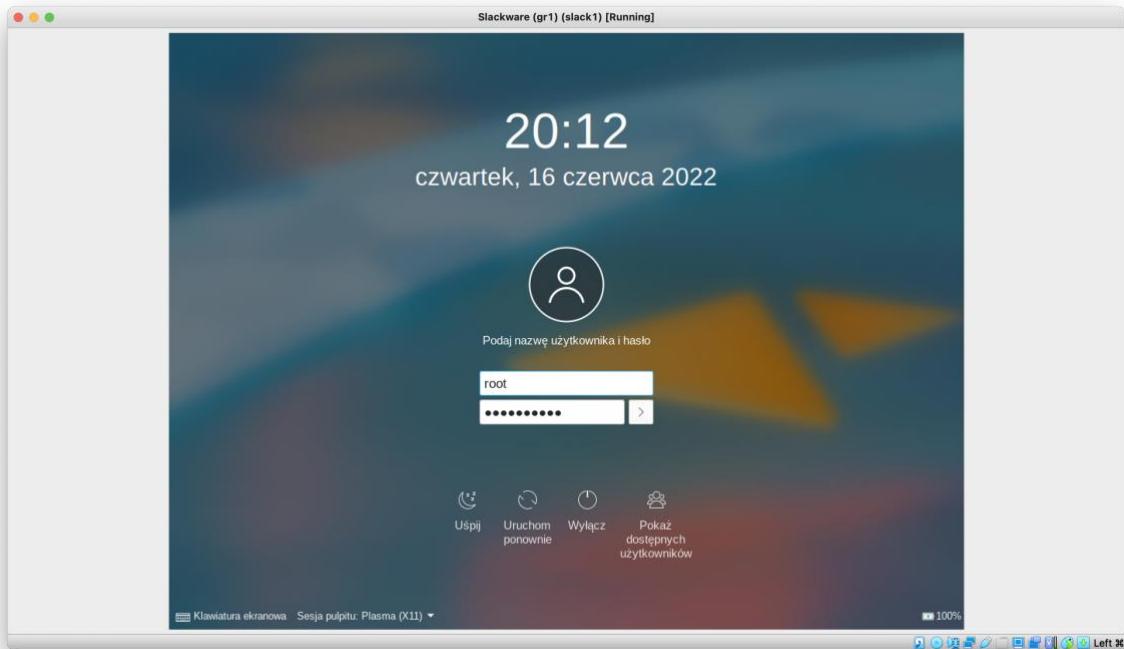
Jak widać na screenach przy uruchomieniu mam opcję **kernel-new** jaką ja poprzednio dodałem do konfiguracji bootloadera.



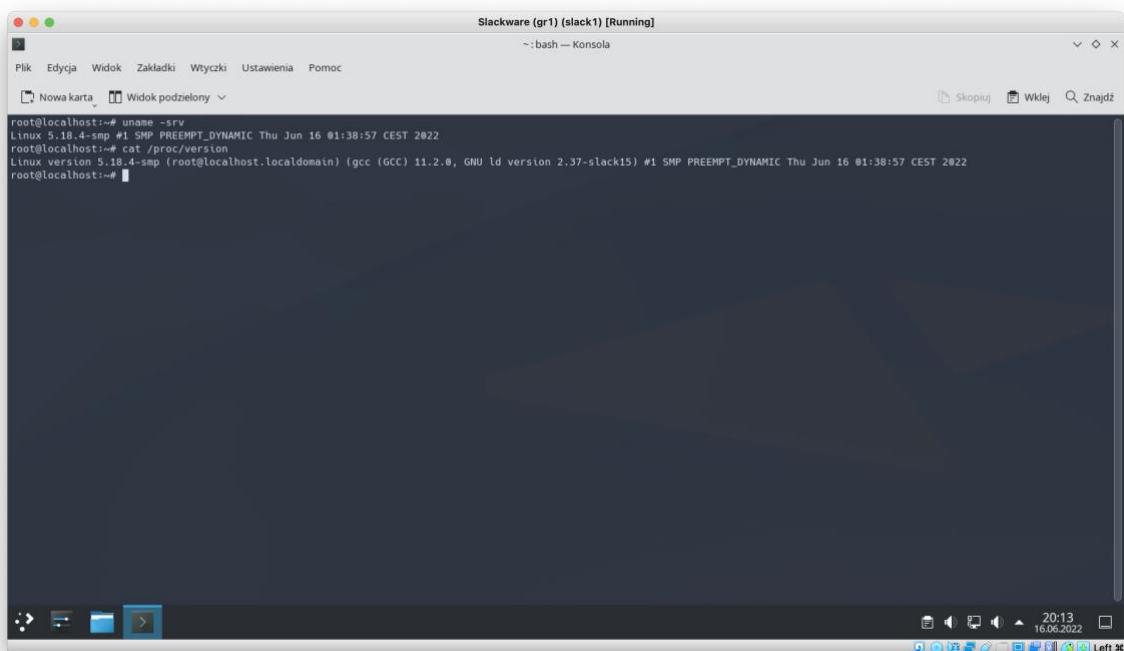
Wybieram ją. System uruchamie się poprawnie.



Loguję się do systemu.



Poleceniami **uname -srv** i **cat /proc/version** sprawdzam wersję jądra.



Własne komentarze do procesu kompilacji

Do procesu kompilacji jądra użyłem Slackware 15 z jądrem Linux 5.15.38 na VirtualBox. Podczas kompilacji nie napotkałem żadnych poważnych problemów, niektóre błędy popełniłem z powodu mojego braku doświadczenia, na przykład nie posiadanie kopii zapasowej poprzedniej konfiguracji jądra, również po raz pierwszy śpieszyłem się i szybko wpisywałem wszystkie polecenia bez zrozumienia co później okazało się problemem. Proces kompilacji okazał się dla mnie bardzo przyjemny i interesujący, w przyszłości spróbuję eksperymentować i robić bardziej wymagające rzeczy.

Metoda **make localmodconfig** dla mnie okazała się nieco „przyjemniejsza”, jak dla początkującego usera Linuxa taka metoda jest bardzo przydatna bo pozwala szybko i bezproblemowo załadować moduły używane w bieżącym systemie, co także znaczy że budowanie jądra zajmuje znacznie mniej czasu . Oczywiście można ręcznie dodać potrzebne moduły które mogą przydać się dla bardziej zaawansowanych userów.