Kompilacja Jądra Linux

Mikołaj Korólczyk

1.Pobieranie najnowszych źródeł kernela.

1. W momencie wykonywania zadania kernel był dostępny w wersji 6.9.3.

```
root@slack64:~# cd /usr/src/
root@slack64:/usr/src# wget https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.9.3.tar.xz
--2024-06-07 13:03:51--  https://cdn.kernel.org/pub/linux/kernel/v6.x/linux-6.9.3.tar.xz
Translacja cdn.kernel.org (cdn.kernel.org)... 2a04:4e42:9::432, 151.101.37.176
Łączenie się z cdn.kernel.org (cdn.kernel.org)|2a04:4e42:9::432|:443... nieudane: Przekroczony czas oczekiwania na p
ołączenie.
Łączenie się z cdn.kernel.org (cdn.kernel.org)|151.101.37.176|:443... połączono.
Żądanie HTTP wysłano, oczekiwanie na odpowiedź... 200 OK
Długość: 144036552 (137M) [application/x-xz]
Zapis do: `linux-6.9.3.tar.xz'

linux-6.9.3.tar.xz           100%[=============================================>] 137,36M  33,7MB/s     w 4,3s

2024-06-07 13:06:07 (32,1 MB/s) - zapisano `linux-6.9.3.tar.xz' [144036552/144036552]

root@slack64:/usr/src#
```

2. Po pobraniu rozpakowałem archiwum ze źródłami kernela za pomocą polecenia tar -xvpf linux-6.9.3.tar.xz

```
linux-6.9.3/virt/kvm/
linux-6.9.3/virt/kvm/Kconfig
linux-6.9.3/virt/kvm/Makefile.kvm
linux-6.9.3/virt/kvm/async_pf.c
linux-6.9.3/virt/kvm/async_pf.h
linux-6.9.3/virt/kvm/binary_stats.c
linux-6.9.3/virt/kvm/coalesced_mmio.c
linux-6.9.3/virt/kvm/coalesced_mmio.h
linux-6.9.3/virt/kvm/dirty_ring.c
linux-6.9.3/virt/kvm/eventfd.c
linux-6.9.3/virt/kvm/guest_memfd.c
linux-6.9.3/virt/kvm/irqchip.c
linux-6.9.3/virt/kvm/kvm_main.c
linux-6.9.3/virt/kvm/kvm_mm.h
linux-6.9.3/virt/kvm/pfncache.c
linux-6.9.3/virt/kvm/vfio.c
linux-6.9.3/virt/kvm/vfio.h
linux-6.9.3/virt/lib/
linux-6.9.3/virt/lib/Kconfig
linux-6.9.3/virt/lib/Makefile
linux-6.9.3/virt/lib/irqbypass.c
root@slack64:/usr/src#
```

3. Następnie przechodzę do katalogu z rozpakowanym kernelem:

```
root@slack64:/usr/src# cd linux-6.9.3
root@slack64:/usr/src/linux-6.9.3#
```

2.Kompilacja jądra metodą „starą”.

1. Rozpoczynam tworzenie pliku konfiguracyjnego:

```
root@slack64:/usr/src/linux-6.9.3# make localmodconfig
```

Polecenie wymaga potwierdzania mnóstwa opcji, więc trzeba wykazać się cierpliwością podczas akceptowania domyślnych wartości proponowanych przez wywołany skrypt:

```
  Test functions located in the hexdump module at runtime (TEST_HEXDUMP) [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
  Test the Maple Tree code at runtime or module load (TEST_MAPLE_TREE) [N/m/y/?] (NEW)
  Perform selftest on resizable hash table (TEST_RHASHTABLE) [N/m/y/?] n
  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 HMM (Heterogeneous Memory Management) (TEST_HMM) [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
  Test module for correctness and stress of objpool (TEST_OBJPOOL) [N/m/?] (NEW)
#
# configuration written to .config
#

root@slack64:/usr/src/linux-6.9.3#
```

Aby sprawdzić aktualnie załadowane moduły, można w tym celu posłużyć się poleceniem lsmod:

```
root@slack64:/usr/src/linux-6.9.3# lsmod
Module                  Size  Used by
fuse                  159744  5
cfg80211             1032192  0
8021q                  40960  0
garp                   16384  1 8021q
mrp                    20480  1 8021q
stp                    16384  1 garp
llc                    16384  2 stp,garp
rfkill                 28672  3 cfg80211
efivarfs               16384  1
ipv6                  593920  60
intel_rapl_msr         20480  0
joydev                 24576  0
intel_rapl_common      28672  1 intel_rapl_msr
vmwgfx                376832  3
crct10dif_pclmul       16384  1
crc32_pclmul           16384  0
ttm                    77824  1 vmwgfx
snd_intel8x0           45056  2
ghash_clmulni_intel    16384  0
```

2. Po wygenerowaniu pliku konfiguracyjnego można przejść do kompilacji obrazu jądra. Aby nieco przyśpieszyć ten proces, do polecenia make dodaję flagę „j” z parametrem „4”. Dzięki temu proces kompilacji wykonywany jest na 4 wątkach procesora.

   make -j 4 bzImage

   ```
   root@slack64:/usr/src/linux-6.9.3# make -j 4 bzImage
   ```

   Powyższe polecenie rozpoczyna proces kompilacji. Można teraz zaparzyć sobie herbatę, posprzątać w mieszkaniu lub wyprowadzić psa na spacer. Wpatrywanie się w okienko terminala nie przyśpieszy procesu kompilacji oraz nie gwarantuje, że zakończy się on bez errorów.

   ```
     CC      arch/x86/boot/video.o
     CC      arch/x86/boot/video-mode.o
     CC      arch/x86/boot/version.o
     LDS     arch/x86/boot/compressed/vmlinux.lds
     CC      arch/x86/boot/video-vga.o
     AS      arch/x86/boot/compressed/kernel_info.o
     CC      arch/x86/boot/video-vesa.o
     CC      arch/x86/boot/video-bios.o
     HOSTCC  arch/x86/boot/tools/build
     AS      arch/x86/boot/compressed/head_64.o
     VOFFSET arch/x86/boot/compressed/../voffset.h
     CPUSTR  arch/x86/boot/cpustr.h
     CC      arch/x86/boot/compressed/string.o
     CC      arch/x86/boot/compressed/cmdline.o
     CC      arch/x86/boot/compressed/error.o
     CC      arch/x86/boot/cpu.o
     OBJCOPY arch/x86/boot/compressed/vmlinux.bin
     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/ident_map_64.o
     CC      arch/x86/boot/compressed/idt_64.o
     AS      arch/x86/boot/compressed/idt_handlers_64.o
     AS      arch/x86/boot/compressed/mem_encrypt.o
     CC      arch/x86/boot/compressed/pgtable_64.o
     CC      arch/x86/boot/compressed/sev.o
     CC      arch/x86/boot/compressed/acpi.o
     CC      arch/x86/boot/compressed/mem.o
     CC      arch/x86/boot/compressed/efi.o
     AS      arch/x86/boot/compressed/efi_mixed.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@slack64:/usr/src/linux-6.9.3#
   ```

3. Obraz jądra skompilowany. Następnie trzeba zbudować moduły jądra poleceniem make modules:

   ```
   root@slack64:/usr/src/linux-6.9.3# make modules
   ```

Wynik budowania modułów:

4. Następnie instaluję moduły za pomocą polecenia

   make modules_install

```
root@slack64:/usr/src/linux-6.9.3# make modules_install
  SYMLINK /lib/modules/6.9.3/build
  INSTALL /lib/modules/6.9.3/modules.order
  INSTALL /lib/modules/6.9.3/modules.builtin
  INSTALL /lib/modules/6.9.3/modules.builtin.modinfo
  INSTALL /lib/modules/6.9.3/kernel/arch/x86/crypto/ghash-clmulni-intel.ko
  INSTALL /lib/modules/6.9.3/kernel/arch/x86/crypto/crc32-pclmul.ko
  INSTALL /lib/modules/6.9.3/kernel/arch/x86/crypto/crct10dif-pclmul.ko
  INSTALL /lib/modules/6.9.3/kernel/fs/fuse/fuse.ko
  INSTALL /lib/modules/6.9.3/kernel/fs/efivarfs/efivarfs.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/acpi/ac.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/acpi/button.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/acpi/video.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/acpi/battery.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/char/agp/agpgart.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/char/agp/intel-agp.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/char/agp/intel-gtt.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/gpu/drm/drm.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/gpu/drm/drm_ttm_helper.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/gpu/drm/drm_kms_helper.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/gpu/drm/ttm/ttm.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/gpu/drm/vmwgfx/vmwgfx.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/block/loop.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/net/ethernet/intel/e1000/e1000.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/usb/host/ehci-hcd.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/usb/host/ehci-pci.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/usb/host/ohci-hcd.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/usb/host/ohci-pci.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/input/serio/serio_raw.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/input/mouse/psmouse.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/input/joydev.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/input/evdev.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/i2c/busses/i2c-piix4.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/i2c/i2c-core.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/platform/x86/dell/dell-wmi-ddv.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/platform/x86/wmi.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/virt/vboxguest/vboxguest.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/virt/coco/tsm.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/virt/coco/sev-guest/sev-guest.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/powercap/intel_rapl_common.ko
  INSTALL /lib/modules/6.9.3/kernel/drivers/powercap/intel_rapl_msr.ko
  INSTALL /lib/modules/6.9.3/kernel/sound/soundcore.ko
  INSTALL /lib/modules/6.9.3/kernel/sound/core/snd.ko
  INSTALL /lib/modules/6.9.3/kernel/sound/core/snd-timer.ko
```

5. Sprawdzam, czy moduły są zainstalowane

   ls /lib/modules/6.9.3/

```
root@slack64:/usr/src/linux-6.9.3# ls /lib/modules/6.9.3/
build          modules.alias.bin          modules.builtin.bin      modules.dep.bin  modules.softdep
kernel         modules.builtin            modules.builtin.modinfo  modules.devname  modules.symbols
modules.alias  modules.builtin.alias.bin  modules.dep              modules.order    modules.symbols.bin
root@slack64:/usr/src/linux-6.9.3#
```

Po procesie kompilacji trzeba sprawdzić, czy system uruchomi się z nowym jądrem. Wymaga to skonfigurowania kilku rzeczy w systemie.

1. Pierwszym krokiem jest przekopiowanie plików kernela do systemu:

```
root@slack64:/usr/src/linux-6.9.3# cp arch/x86_64/boot/bzImage /boot/vmlinuz-custom-6.9.3
root@slack64:/usr/src/linux-6.9.3# cp System.map /boot/System.map-custom-6.9.3
root@slack64:/usr/src/linux-6.9.3# cp .config /boot/config-custom-6.9.3
root@slack64:/usr/src/linux-6.9.3#
```

2. Następnie tworzę link symboliczny do tablicy symboli kernela:

```
root@slack64:/usr/src/linux-6.9.3# cd /boot/
root@slack64:/boot# rm System.map
root@slack64:/boot# ln -s System.map-custom-6.9.3 System.map
root@slack64:/boot#
```

3. Tworzę dysk ram:

```
root@slack64:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 6.9.3
#
# 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 6.9.3 -f ext4 -r /dev/sda3 -m ext4 -u -o /boot/initrd.gz
root@slack64:/boot# mkinitrd -c -k 6.9.3 -f ext4 -r /dev/sda3 -m ext4 -u -o /boot/initrd-custom-6.9.3.gz
51365 bloków
/boot/initrd-custom-6.9.3.gz created.
Be sure to run lilo again if you use it.
root@slack64:/boot#
```

4. Ponownie konfiguruję GRUB:

```
root@slack64:/boot# grub-mkconfig -o /boot/grub/grub.cfg
Generowanie pliku konfiguracyjnego gruba...
Znaleziono obraz Linuksa: /boot/vmlinuz-huge-5.15.19
Znaleziono obraz initrd: /boot/initrd.gz
Znaleziono obraz Linuksa: /boot/vmlinuz-huge
Znaleziono obraz initrd: /boot/initrd.gz
Znaleziono obraz Linuksa: /boot/vmlinuz-generic-5.15.19
Znaleziono obraz initrd: /boot/initrd.gz
Znaleziono obraz Linuksa: /boot/vmlinuz-generic
Znaleziono obraz initrd: /boot/initrd.gz
Znaleziono obraz Linuksa: /boot/vmlinuz-custom-6.9.3
Znaleziono obraz initrd: /boot/initrd.gz
Uwaga: os-prober nie zostanie uruchomiony w celu wykrycia innych uruchamialnych partycji.
Systemy na nich nie zostaną dodane do konfiguracji rozruchowej GRUB-a.
Proszę sprawdzić dokumentację dotyczącą GRUB_DISABLE_OS_PROBER.
Dodawanie wpisu menu rozruchowego dla ustawień firmware'u UEFI...
gotowe
root@slack64:/boot#
```

5. Kopiuję obraz jądra oraz ramdisku na partycję EFI:

```
root@slack64:/boot# cp /boot/vmlinuz-custom-6.9.3 /boot/efi/EFI/Slackware/.
root@slack64:/boot# cp /boot/initrd-custom-6.9.3.gz /boot/efi/EFI/Slackware/.
root@slack64:/boot#
```

6. Dodaję nowy wpisu do konfiguracji bootloadera elilo:

```
root@slack64:/boot# vi /boot/efi/EFI/Slackware/elilo.conf
```

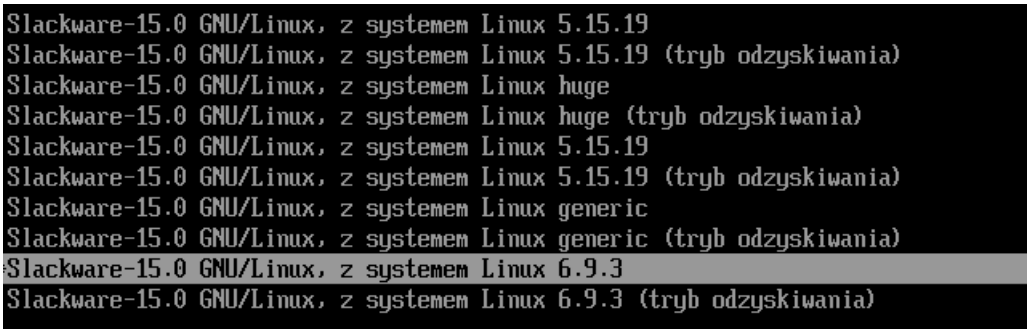
7. Plik konfiguracyjny po dodaniu wpisu:

```
prompt
delay=1
timeout=30
default=custom-kernel-6.9.3
#
image=vmlinuz
        label=Slackware-15.0
        initrd=initrd.gz
        read-only
        append="root=/dev/sda3 vga=normal ro"

image=vmlinuz-custom-6.9.3
        label=custom-kernel-6.9.3
        initrd=initrd-custom-6.9.3.gz
        read-only
        append="root=/dev/sda3 vga=normal ro"
```

Był to ostatni etap konfiguracji. Wykonuję migawkę w VirtualBox i restartuję system.

Po ponownym uruchomieniu w GRUB dostępna jest rozruchu systemu z nowo zainstalowanym jądrem:



System uruchamia się prawidłowo. Sprawdzam wersję jądra:

```
student@slack64:~$ uname -r
6.9.3
student@slack64:~$
```

3. Kompilacja jądra metodą „nową”.

1. Rozpoczynam tworzenie nowego pliku konfiguracyjnego:

```
root@slack64:/usr/src/linux-6.9.3# cp /boot/config .config
root@slack64:/usr/src/linux-6.9.3# ./scripts/kconfig/streamline_config.pl > config_strip
using config: '.config'
module fb_sys_fops did not have configs CONFIG_FB_SYSMEM_FOPS
root@slack64:/usr/src/linux-6.9.3# mv .config config.bak
root@slack64:/usr/src/linux-6.9.3# mv config_strip .config
root@slack64:/usr/src/linux-6.9.3# make oldconfig
```

Skrypt wygenerował plik konfiguracyjny:

```
  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 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
  Test the Maple Tree code at runtime or module load (TEST_MAPLE_TREE) [N/m/y/?] (NEW)
  Perform selftest on resizable hash table (TEST_RHASHTABLE) [N/m/y/?] n
  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 HMM (Heterogeneous Memory Management) (TEST_HMM) [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
  Test module for correctness and stress of objpool (TEST_OBJPOOL) [N/m/?] (NEW)
#
# configuration written to .config
#

root@slack64:/usr/src/linux-6.9.3#
```

2. Kompiluje jądro poleceniem make -j 4 bzImage:

```
  CC      arch/x86/boot/compressed/idt_64.o
  AS      arch/x86/boot/compressed/idt_handlers_64.o
  AS      arch/x86/boot/compressed/mem_encrypt.o
  CC      arch/x86/boot/compressed/pgtable_64.o
  CC      arch/x86/boot/compressed/sev.o
  CC      arch/x86/boot/compressed/acpi.o
  CC      arch/x86/boot/compressed/mem.o
  CC      arch/x86/boot/compressed/efi.o
  AS      arch/x86/boot/compressed/efi_mixed.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@slack64:/usr/src/linux-6.9.3#
```

3. Buduję moduły poleceniem make -j 4 modules:

```
  LD [M]  drivers/i2c/i2c-core.ko
  LD [M]  drivers/virt/vboxguest/vboxguest.ko
  LD [M]  drivers/virt/coco/tsm.ko
  LD [M]  drivers/powercap/intel_rapl_common.ko
  LD [M]  drivers/virt/coco/sev-guest/sev-guest.ko
  LD [M]  sound/core/snd.ko
  LD [M]  drivers/powercap/intel_rapl_msr.ko
  LD [M]  sound/soundcore.ko
  LD [M]  sound/core/snd-timer.ko
  LD [M]  sound/core/snd-pcm.ko
  LD [M]  sound/pci/ac97/snd-ac97-codec.ko
  LD [M]  sound/pci/snd-intel8x0.ko
  LD [M]  sound/ac97_bus.ko
  LD [M]  net/802/p8022.ko
  LD [M]  net/802/psnap.ko
  LD [M]  net/802/stp.ko
  LD [M]  net/802/garp.ko
  LD [M]  net/802/mrp.ko
  LD [M]  net/wireless/cfg80211.ko
  LD [M]  net/ipv6/ipv6.ko
  LD [M]  net/llc/llc.ko
  LD [M]  net/rfkill/rfkill.ko
  LD [M]  net/8021q/8021q.ko
root@slack64:/usr/src/linux-6.9.3#
```

4. instaluję moduły poleceniem make modules_install:

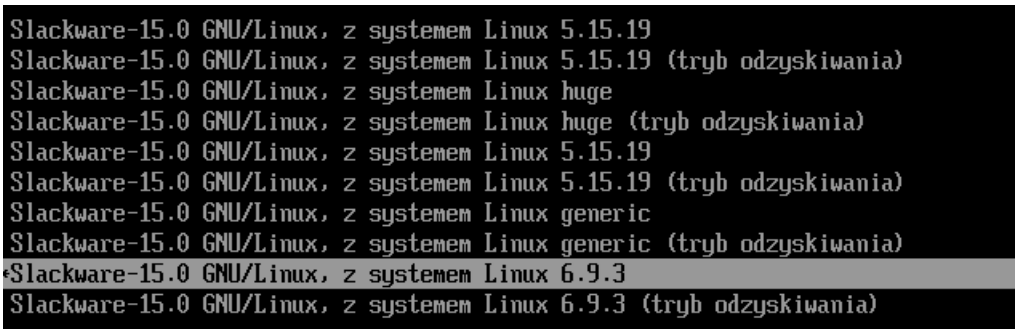
```
  INSTALL /lib/modules/6.9.3/kernel/sound/core/snd-pcm.ko
  INSTALL /lib/modules/6.9.3/kernel/sound/pci/snd-intel8x0.ko
  INSTALL /lib/modules/6.9.3/kernel/sound/pci/ac97/snd-ac97-codec.ko
  INSTALL /lib/modules/6.9.3/kernel/sound/ac97_bus.ko
  INSTALL /lib/modules/6.9.3/kernel/net/802/p8022.ko
  INSTALL /lib/modules/6.9.3/kernel/net/802/psnap.ko
  INSTALL /lib/modules/6.9.3/kernel/net/802/stp.ko
  INSTALL /lib/modules/6.9.3/kernel/net/802/garp.ko
  INSTALL /lib/modules/6.9.3/kernel/net/802/mrp.ko
  INSTALL /lib/modules/6.9.3/kernel/net/ipv6/ipv6.ko
  INSTALL /lib/modules/6.9.3/kernel/net/8021q/8021q.ko
  INSTALL /lib/modules/6.9.3/kernel/net/wireless/cfg80211.ko
  INSTALL /lib/modules/6.9.3/kernel/net/llc/llc.ko
  INSTALL /lib/modules/6.9.3/kernel/net/rfkill/rfkill.ko
  DEPMOD  /lib/modules/6.9.3
root@slack64:/usr/src/linux-6.9.3#
```

5. sprawdzam, czy moduły zostały zainstalowane:

```
root@slack64:/usr/src/linux-6.9.3# ls /lib/modules/6.9.3/
build          modules.alias.bin          modules.builtin.bin      modules.dep.bin  modules.softdep
kernel         modules.builtin            modules.builtin.modinfo  modules.devname  modules.symbols
modules.alias  modules.builtin.alias.bin  modules.dep              modules.order    modules.symbols.bin
root@slack64:/usr/src/linux-6.9.3#
```

Dalsze kroki dotyczą konfiguracji systemu, aby dało się go uruchomić z nowym kernelem. Etapy konfiguracji są identyczne z etapami w punkcie drugim, przy instalacji z konfiguracją wygenerowaną starą metodą, więc nie będę ich dokumentował w tej części raportu.

Tym razem również nowa wersja dostępna jest w GRUB i da się ją uruchomić:



4. Podsumowanie.

Proces kompilacji jądra nie przysporzył mi większych kłopotów zarówno dla metody starej, jak i nowej. Udało mi się również uruchomić system z nowym kernelem z konfiguracją wygenerowaną na dwa sposoby. Systemy uruchomione z tymi kernelami zdawały się funkcjonować poprawnie. System uruchamiał się, środowisko graficzne również, interfejsy sieciowe funkcjonowały poprawnie, a programy typu firefox, zip, dolphin działały bez zarzutu. Nie zauważyłem wyraźnych różnic w czasie rozruchu systemu między tymi kernelami.

Mimo iż nie zauważyłem wyraźnych różnic w działaniu tych kerneli, to istnieje dość istotna różnica między konfiguracjami kerneli wygenerowanych przy użyciu tych dwóch metod. Po krótkim researchu udało mi się ustalić, że:

- localmodconfig – wykrywa obecnie używane moduły i przygotowuje minimalną konfigurację niezbędną do obsługi tych modułów. Skraca to znacznie czas kompilacji, ale przy zmianach sprzętowych system może mieć problem z obsługą nowych urządzeń, które wymagają niezaładowanych modułów.
- streamline_config.pl – działa podobnie jak localmodconfig z tą różnicą, że oprócz uwzględnienia tylko obecnie używanych modułów śledzi również zależności między nimi i pozwala załadować moduły obecnie niewykrywane (lsmods), jeśli są wymagane przez inne moduły. Dzięki takiemu podejściu system powinien zachowywać się bardziej stabilnie mimo zmian w sprzęcie kosztem nieco dłuższego czasu kompilacji.