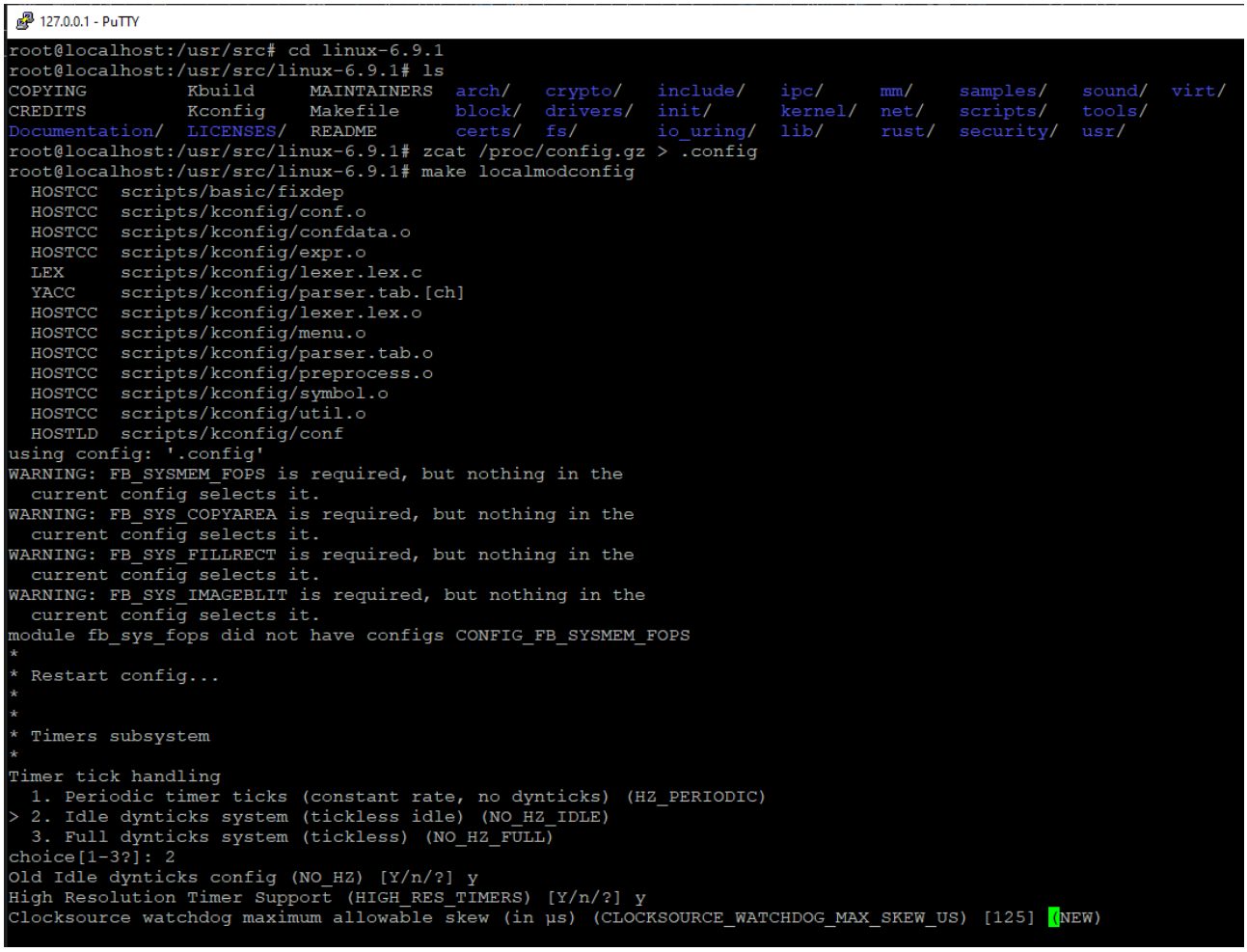
Kompilacja Jądra Linux

1. Bartosz Szynkaruk

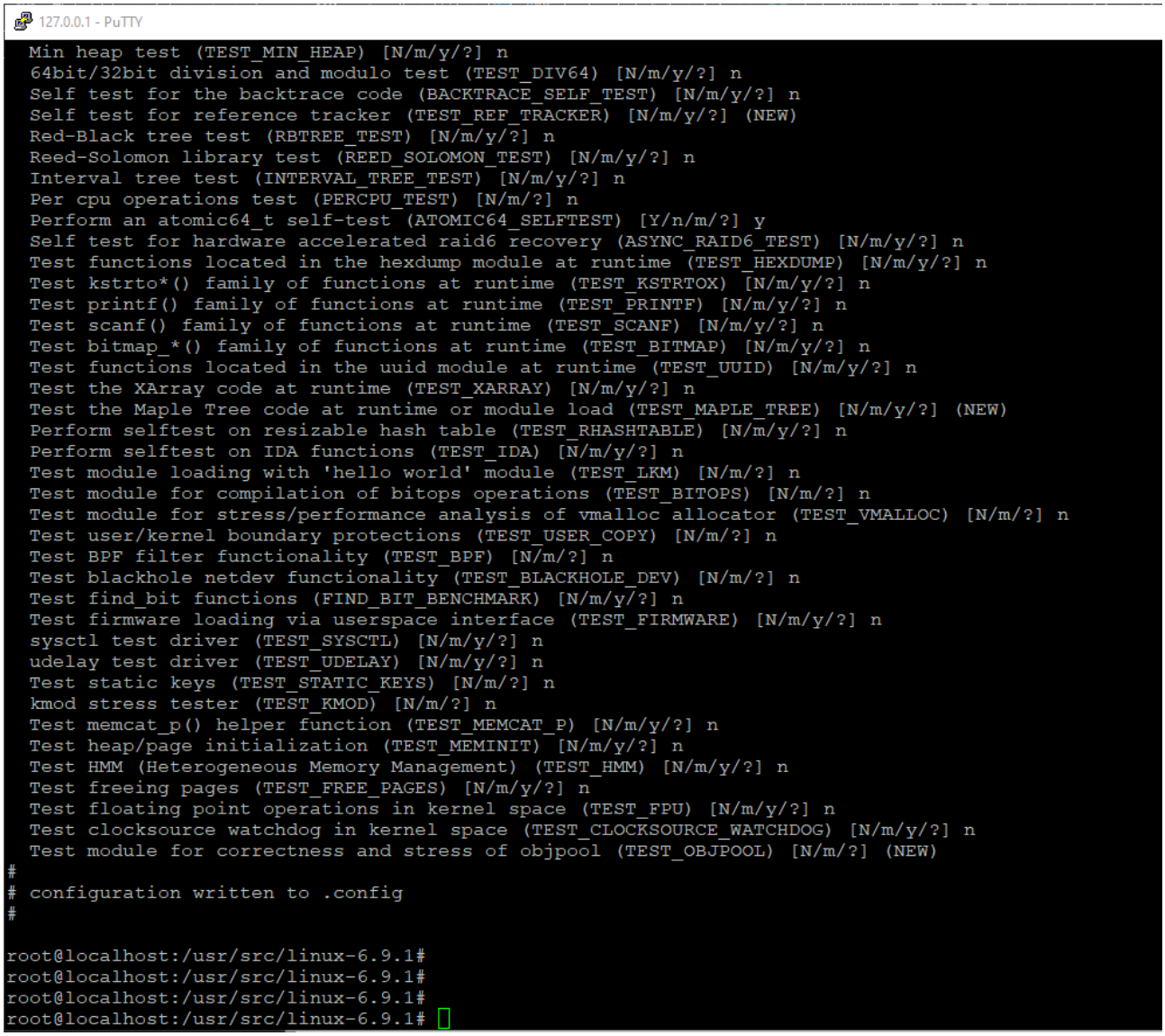
2. Przebieg procesu kompilacji dla metody starej

Proces kompilacji przy użyciu starej metody rozpocząłem od przejścia do katalogu /usr/src. Usunąłem pliki które zostały po zajęciach, czyli cały katalog linux-6.9.1 oraz na nowo rozpakowałem plik linux-6.9.1.tar.xz, i skopiowałem tam starą konfigurację z /proc/config.gz do .config.

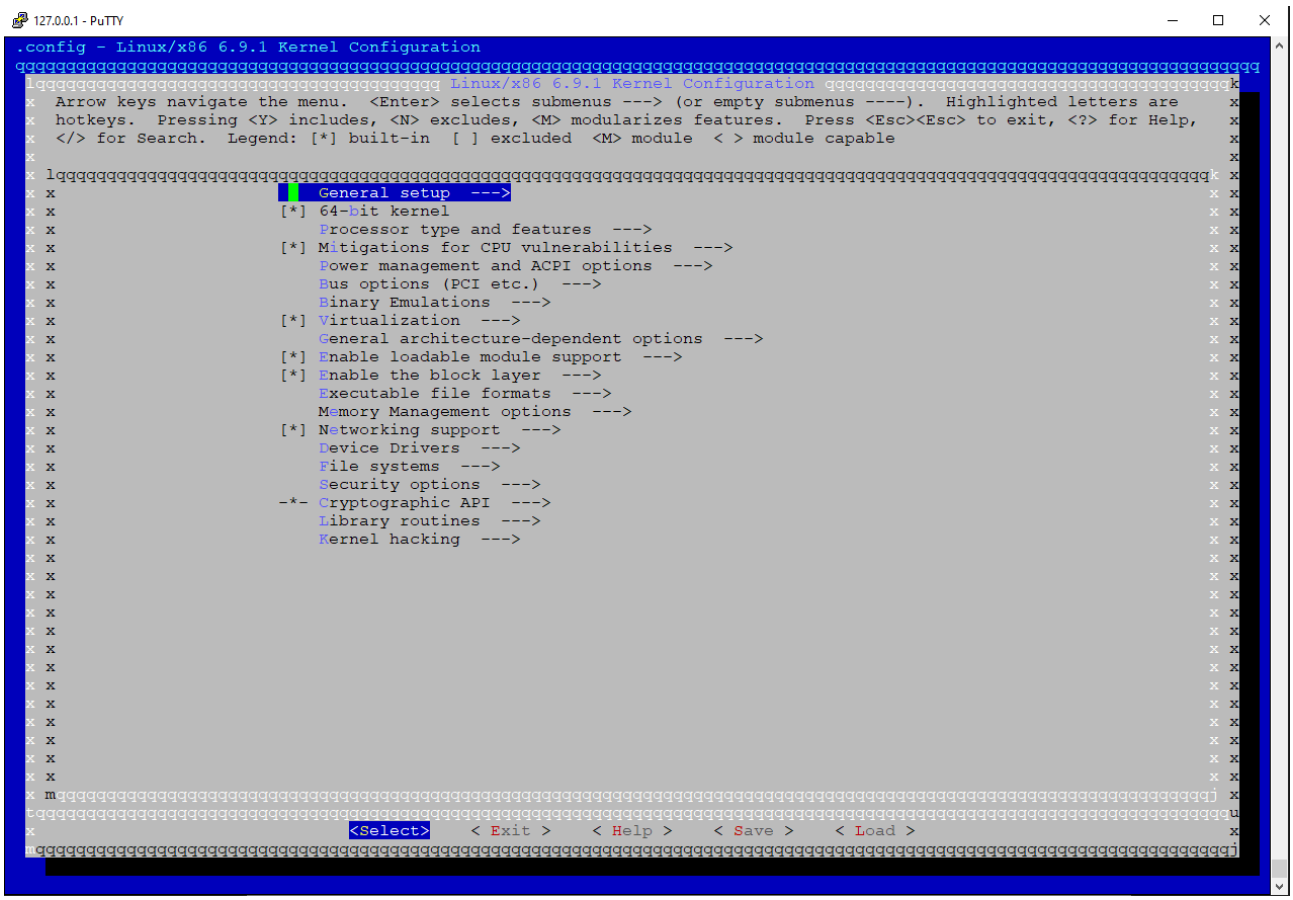
Następnie użyłem polecenia **make localmodconfig**



Następnie trzymałem przycisk enter aby zakończyć proces konfiguracji.



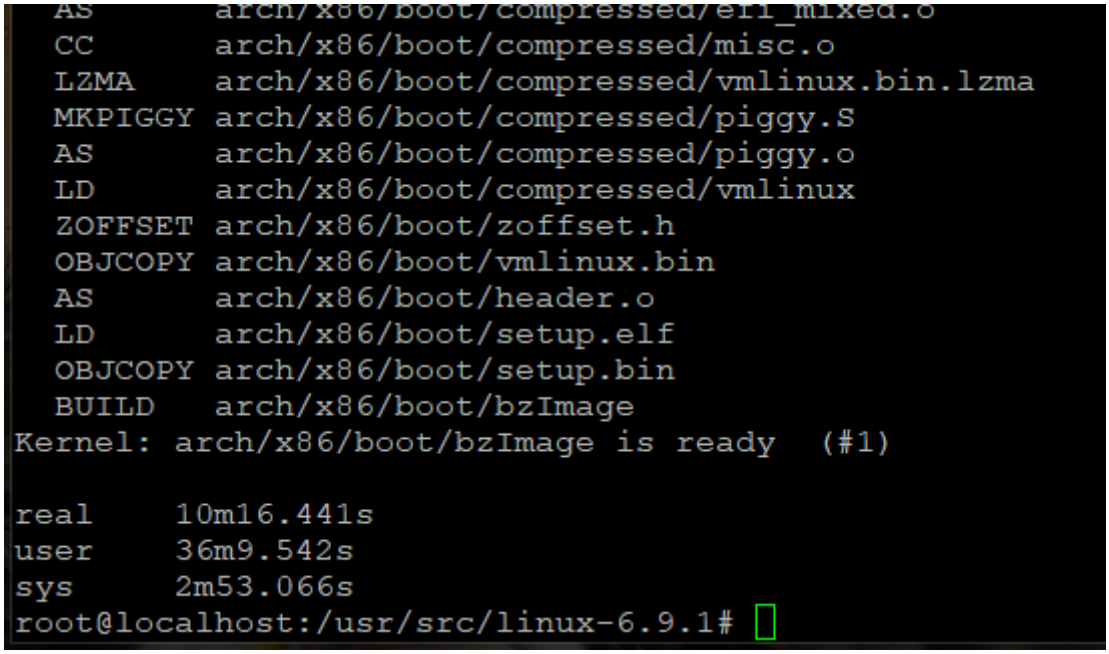
Jak widać, konfiguracja przebiegła pomyślnie.
W następnej kolejności użyłem polecenia **make menuconfig**.



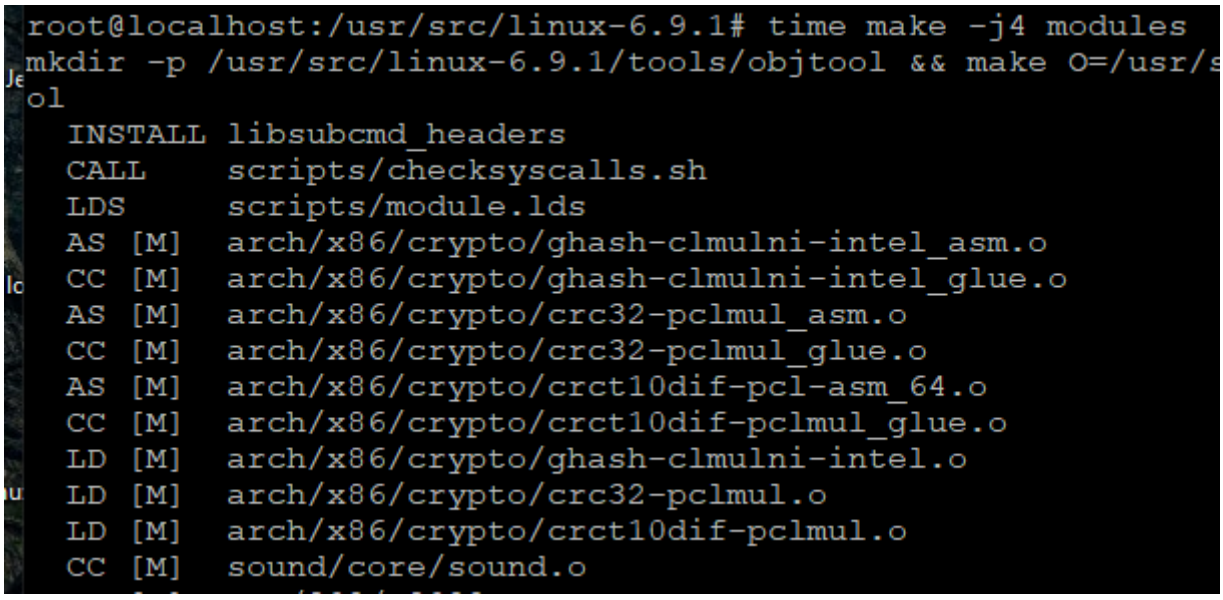
Tekstowe menu konfiguracji wyświetliło się pomyślnie, jednak nie wprowadzałem w nim żadnych zmian. Następnym krokiem jest kompilacja jądra. Użyłem polecenia **time make -j4 bzImage** żeby dodatkowo zmierzyć czas kompilacji. Moja maszyna wirtualna używa 4 rdzeni, stąd argument -j4.

```
root@localhost:/usr/src/linux-6.9.1# time make -j4 bzImage
  WRAP    arch/x86/include/generated/uapi/asm/bpf_perf_event.h
  GEN     arch/x86/include/generated/asm/orc_hash.h
  WRAP    arch/x86/include/generated/uapi/asm/errno.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_32.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
  WRAP    arch/x86/include/generated/uapi/asm/resource.h
  HOSTCC  arch/x86/tools/relocs_32.o
  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
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_64.h
  SYSHDR  arch/x86/include/generated/uapi/asm/unistd_x32.h
  UPD     include/config/kernel.release
  WRAP    arch/x86/include/generated/asm/early_ioremap.h
```

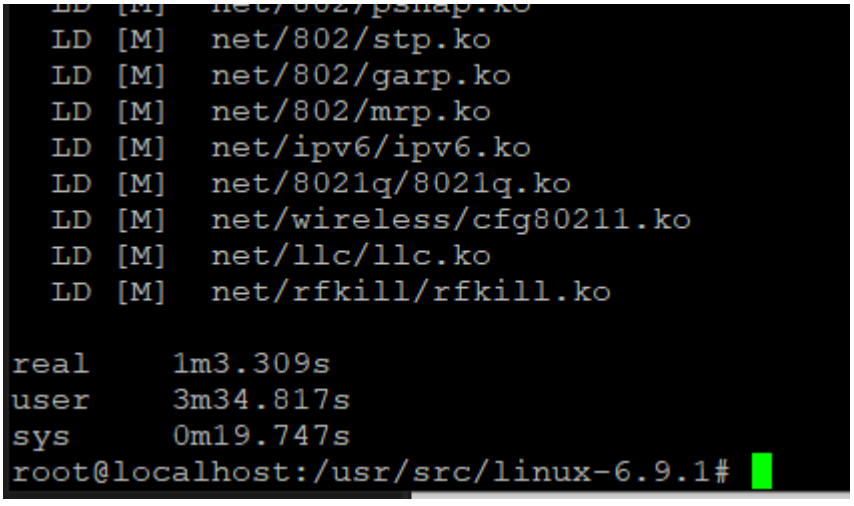
Wyniki kompilacji obrazu jądra wraz z wynikiem polecenia time:



Następnie użyłem polecenia **time make -j4 modules**.



Oto wynik:



Następne polecenie to **time make -j4 modules_install**

```
root@localhost:/usr/src/linux-6.9.1# time make -j4 modules_install
  INSTALL /lib/modules/6.9.1/modules.order
  INSTALL /lib/modules/6.9.1/modules.builtin
  INSTALL /lib/modules/6.9.1/modules.builtin.modinfo
  SYMLINK /lib/modules/6.9.1/build
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/crypto/ghash-clmulni-intel.ko
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/crypto/crc32-pclmul.ko
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/crypto/crct10dif-pclmul.ko
  INSTALL /lib/modules/6.9.1/kernel/fs/efivarfs/efivarfs.ko
  INSTALL /lib/modules/6.9.1/kernel/drivers/acpi/ac.ko
  INSTALL /lib/modules/6.9.1/kernel/drivers/acpi/button.ko
  INSTALL /lib/modules/6.9.1/kernel/drivers/acpi/video.ko
  INSTALL /lib/modules/6.9.1/kernel/drivers/char/agp/agpgart.ko
```

Polecenie wykonało się bardzo szybko.

```
  INSTALL /lib/modules/6.9.1/kernel/net/802/garp.ko
  INSTALL /lib/modules/6.9.1/kernel/net/802/mrp.ko
  INSTALL /lib/modules/6.9.1/kernel/net/ipv6/ipv6.ko
  INSTALL /lib/modules/6.9.1/kernel/net/8021q/8021q.ko
  INSTALL /lib/modules/6.9.1/kernel/net/wireless/cfg80211.ko
  INSTALL /lib/modules/6.9.1/kernel/net/llc/llc.ko
  INSTALL /lib/modules/6.9.1/kernel/net/rfkill/rfkill.ko
  DEPMOD  /lib/modules/6.9.1

real    0m0.648s
user    0m0.504s
sys     0m0.483s
root@localhost:/usr/src/linux-6.9.1#
```

Skompilowane jądro, mapę symboli oraz konfigurację skopiowałem do katalogu /boot.

```
127.0.0.1 - PuTTY
root@localhost:/usr/src/linux-6.9.1# cp arch/x86_64/boot/bzImage /boot/vmlinuz-old-custom-6.9.1
root@localhost:/usr/src/linux-6.9.1# cp System.map /boot/System.map-old-custom-6.9.1
root@localhost:/usr/src/linux-6.9.1# cp .config /boot/config-old-custom-6.9.1
root@localhost:/usr/src/linux-6.9.1#
```

Następnie utworzyłem link do nowej mapy symboli.

```
root@localhost:/boot# rm System.map
root@localhost:/boot# ln -s System.map-old-custom-6.9.1 System.map
```

Oraz wygenerowałem ramdisk dla nowego kernela.

```
root@localhost:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 6.9.1
#
# 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 6.9.1 -f ext4 -r /dev/sda3 -m ext4 -u -o /boot/initrd.gz
root@localhost:/boot# ^C
root@localhost:/boot# mkinitrd -c -k 6.9.1 -f ext4 -r /dev/sda3 -m ext4 -u -o /boot/initrd-old-custom-6.9.1
51364 blocks
/boot/initrd-old-custom-6.9.1 created.
Be sure to run lilo again if you use it.
root@localhost:/boot#
```

I zaktualizowałem konfigurację bootloadera grub.

```
root@localhost:~# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-old-custom-6.9.1
Found initrd image: /boot/initrd.gz
Found linux image: /boot/vmlinuz-huge-5.15.19
Found initrd image: /boot/initrd.gz
Found linux image: /boot/vmlinuz-huge
Found initrd image: /boot/initrd.gz
Found linux image: /boot/vmlinuz-generic-5.15.19
Found initrd image: /boot/initrd.gz
Found linux image: /boot/vmlinuz-generic
Found initrd image: /boot/initrd.gz
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
root@localhost:~#
```

Po restarcie systemu z nowym jądrem:

```
127.0.0.1 - PuTTY
login as: root
Keyboard-interactive authentication prompts from server:
| Password:
End of keyboard-interactive prompts from server
Last login: Sun Jun 16 16:45:04 2024
Linux 6.9.1.
root@localhost:~# uname -r
6.9.1
root@localhost:~#
```

Wykonałem kilka testów wydajności nowego kernela. Na początek zmierzyłem czas startu systemu, od momentu wciśnięcia klawisza enter w menu grub do wyświetlenia linijki „slack login”.
Wykonałem 3 próby.
1) 13.26 s
2) 11.07 s
3) 14.73 s

Średnia: 13.02 s

3. Przebieg procesu kompilacji dla metody nowej

W tym przypadku wykonałem te same kroki co wcześniej – usunąłem folder linux-6.9.1, na nowo rozpakowałem archiwum oraz skopiowałem starą konfigurację.

Zgodnie z instrukcją znajdującą się w skrypcie:

```
#
# Howto:
#
#  1. Boot up the kernel that you want to stream line 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"
#
```

Uruchomiłem skrypt: **scripts/kconfig/streamline_config.pl** zapisując konfigurację od razu do .config, bez tworzenia kopii.

```
root@localhost:/usr/src/linux-6.9.1# scripts/kconfig/streamline_config.pl > .config
using config: '.config'
module vboxguest did not have configs CONFIG_VBOXGUEST
module snd_timer did not have configs CONFIG_SND_TIMER
module psmouse did not have configs CONFIG_MOUSE_PS2
module serio_raw did not have configs CONFIG_SERIO_RAW
module syscopyarea did not have configs CONFIG_FB_SYS_COPYAREA
module mrp did not have configs CONFIG_MRP
module intel_agp did not have configs CONFIG_AGP_INTEL
module intel_rapl_msr did not have configs CONFIG_INTEL_RAPL
module rfkill did not have configs CONFIG_RFKILL
module snd did not have configs CONFIG_SND
module efivarfs did not have configs CONFIG_EFIVAR_FS
module ipv6 did not have configs CONFIG_IPV6
module sysimgblt did not have configs CONFIG_FB_SYS_IMAGEBLIT
module cfg80211 did not have configs CONFIG_CFG80211
module joydev did not have configs CONFIG_INPUT_JOYDEV
module ac did not have configs CONFIG_ACPI_AC
module stp did not have configs CONFIG_STP
module agpgart did not have configs CONFIG_AGP
module i2c_piix4 did not have configs CONFIG_I2C_PIIX4
module garp did not have configs CONFIG_GARP
```

Skrypt zakończył działanie:

```
module i2c_core did not have configs CONFIG_I2C
module ohci_hcd did not have configs CONFIG_USB_OHCI_HCD
module intel_rapl_common did not have configs CONFIG_INTEL_RAPL_CORE
module ttm did not have configs CONFIG_DRM_TTM
module evdev did not have configs CONFIG_INPUT_EVDEV
module drm_kms_helper did not have configs CONFIG_DRM_KMS_HELPER
module ehci_pci did not have configs CONFIG_USB_EHCI_PCI
module crc32_pclmul did not have configs CONFIG_CRYPTO_CRC32_PCLMUL
root@localhost:/usr/src/linux-6.9.1#
```

Uruchomiłem polecenie **make oldconfig**.

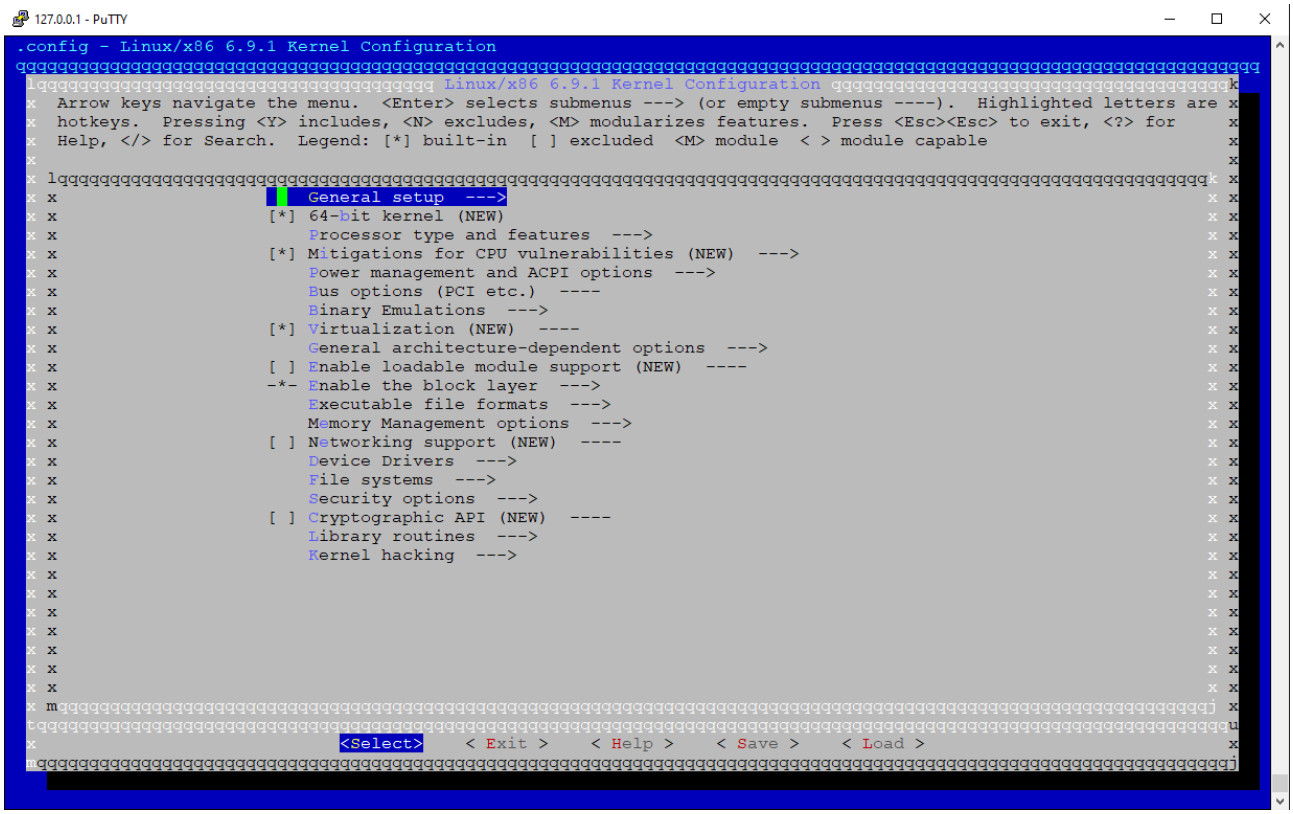
```
root@localhost:/usr/src/linux-6.9.1# make oldconfig
  HOSTCC  scripts/basic/fixdep
  HOSTCC  scripts/kconfig/conf.o
  HOSTCC  scripts/kconfig/confdata.o
  HOSTCC  scripts/kconfig/expr.o
  LEX     scripts/kconfig/lexer.lex.c
  YACC    scripts/kconfig/parser.tab.[ch]
  HOSTCC  scripts/kconfig/lexer.lex.o
  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
.config:429:warning: symbol value 'm' invalid for I8K
.config:3072:warning: symbol value 'm' invalid for NET_DSA_REALTEK_SMI
.config:7237:warning: symbol value 'm' invalid for USB_FOTG210_HCD
.config:7880:warning: symbol value 'm' invalid for VFIO_VIRQFD
.config:8028:warning: symbol value 'm' invalid for VIDEO_ZORAN_DC30
.config:8029:warning: symbol value 'm' invalid for VIDEO_ZORAN_ZR36060
.config:8030:warning: symbol value 'm' invalid for VIDEO_ZORAN_BUZ
.config:8031:warning: symbol value 'm' invalid for VIDEO_ZORAN_DC10
.config:8032:warning: symbol value 'm' invalid for VIDEO_ZORAN_LML33
.config:8033:warning: symbol value 'm' invalid for VIDEO_ZORAN_LML33R10
.config:8034:warning: symbol value 'm' invalid for VIDEO_ZORAN_AVS6EYES
.config:9151:warning: symbol value 'm' invalid for FSCACHE
.config:9587:warning: symbol value 'm' invalid for CRYPTO_BLAKE2S_X86
.config:9685:warning: symbol value 'm' invalid for CRYPTO_ARCH_HAVE_LIB_BLAKE2S
.config:9686:warning: symbol value 'm' invalid for CRYPTO_LIB_BLAKE2S_GENERIC
*
* 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)
  3. Full dynticks system (tickless) (NO_HZ_FULL)
choice[1-3?]: 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) [125] (NEW)
```

Ponownie trzymałem wciśnięty przycisk enter aż do końca konfiguracji.

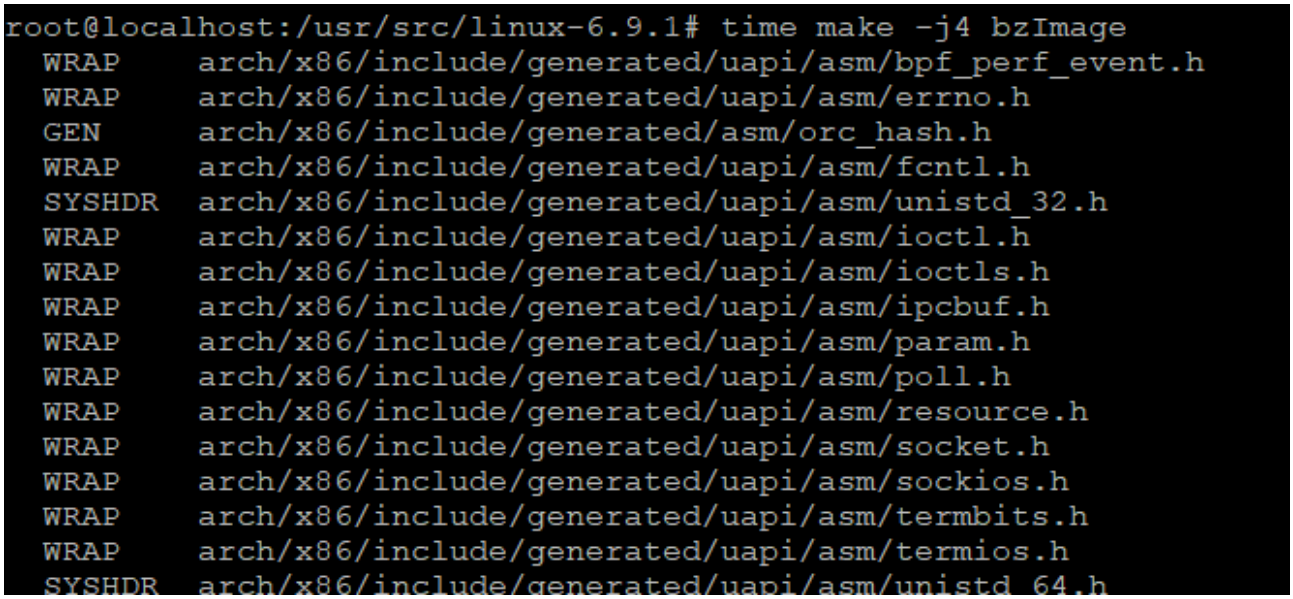
```
  Perform selftest on priority array manager (TEST_PARMAN) [N/m/?] n
  Test module loading with 'hello world' module (TEST_LKM) [M/n/?] m
  Test module for compilation of bitops operations (TEST_BITOPS) [N/m/?] n
  Test module for stress/performance analysis of vmalloc allocator (TEST_VMALLOC) [N/m/?] n
  Test user/kernel boundary protections (TEST_USER_COPY) [M/n/?] m
  Test BPF filter functionality (TEST_BPF) [M/n/?] m
  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) [M/n/y/?] m
  sysctl test driver (TEST_SYSCTL) [N/m/y/?] n
  udelay test driver (TEST_UDELAY) [M/n/y/?] m
  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
  Perform selftest on object aggreration manager (TEST_OBJAGG) [N/m/?] 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@localhost:/usr/src/linux-6.9.1#
root@localhost:/usr/src/linux-6.9.1#
```

Ponownie uruchomiłem menu konfiguracyjne używając **make menuconfig**.



Po wyjściu nie zauważyłem żadnych błędów, więc rozpocząłem kompilację obrazu jądra.



Kompilacja zakończyła się. Oto wyniki:

```
  AS      arch/x86/boot/compressed/piggy.o
  LD      arch/x86/boot/compressed/vmlinux
  ZOFFSET arch/x86/boot/zoffset.h
  OBJCOPY arch/x86/boot/vmlinux.bin
  AS      arch/x86/boot/header.o
  LD      arch/x86/boot/setup.elf
  OBJCOPY arch/x86/boot/setup.bin
  BUILD   arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready  (#1)

real    10m19.917s
user    35m48.757s
sys     3m1.533s
root@localhost:/usr/src/linux-6.9.1#
```

Teraz kompilacja modułów.

```
root@localhost:/usr/src/linux-6.9.1# time make -j4 modules
mkdir -p /usr/src/linux-6.9.1/tools/objtool && make O=/usr/src/linux-6.
  INSTALL libsubcmd_headers
  CALL    scripts/checksyscalls.sh
  LDS     scripts/module.lds
  CC [M]  arch/x86/events/amd/power.o
  CC [M]  mm/hwpoison-inject.o
  CC [M]  arch/x86/kernel/cpu/mce/inject.o
  CC [M]  kernel/time/test_udelay.o
  CC [M]  arch/x86/events/intel/cstate.o
  CC [M]  kernel/trace/ring_buffer_benchmark.o
  CC [M]  fs/quota/quota_v1.o
  LD [M]  arch/x86/kernel/cpu/mce/mce-inject.o
  LD [M]  arch/x86/events/intel/intel-cstate.o
  CC [M]  arch/x86/events/rapl.o
  AS [M]  arch/x86/crypto/twofish-x86_64-asm_64-3way.o
  CC [M]  arch/x86/crypto/twofish_glue_3way.o
```

Proces zakończył się. W tym przypadku kompilacja modułów trwała dużo dłużej.

```
  LD [M]  net/nsh/nsh.ko
  LD [M]  net/vmw_vsock/vsock_loopback.ko
  LD [M]  net/hsr/hsr.ko
  LD [M]  net/qrtr/qrtr.ko
  LD [M]  net/qrtr/qrtr-tun.ko
  LD [M]  net/qrtr/qrtr-smd.ko
  LD [M]  net/qrtr/qrtr-mhi.ko

real    39m9.744s
user    139m56.964s
sys     13m41.264s
root@localhost:/usr/src/linux-6.9.1#
```

Kolejnym krokiem jest instalacja:

```
root@localhost:/usr/src/linux-6.9.1# time make -j4 modules_install
  INSTALL /lib/modules/6.9.1/modules.order
  INSTALL /lib/modules/6.9.1/modules.builtin
  INSTALL /lib/modules/6.9.1/modules.builtin.modinfo
  SYMLINK /lib/modules/6.9.1/build
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/events/intel/intel-cstate.ko
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/events/amd/power.ko
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/events/rapl.ko
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/kernel/cpu/mce/mce-inject.ko
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/crypto/twofish-x86_64-3way.ko
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/crypto/twofish-avx-x86_64.ko
  INSTALL /lib/modules/6.9.1/kernel/arch/x86/crypto/serpent-avx-x86_64.ko
```

```
  INSTALL /lib/modules/6.9.1/kernel/net/vmw_vsock/vsock_loopback.ko
  INSTALL /lib/modules/6.9.1/kernel/net/nsh/nsh.ko
  INSTALL /lib/modules/6.9.1/kernel/net/hsr/hsr.ko
  INSTALL /lib/modules/6.9.1/kernel/net/qrtr/qrtr.ko
  INSTALL /lib/modules/6.9.1/kernel/net/qrtr/qrtr-smd.ko
  INSTALL /lib/modules/6.9.1/kernel/net/qrtr/qrtr-tun.ko
  INSTALL /lib/modules/6.9.1/kernel/net/qrtr/qrtr-mhi.ko
  DEPMOD  /lib/modules/6.9.1

real    0m12.933s
user    0m29.556s
sys     0m12.725s
root@localhost:/usr/src/linux-6.9.1#
```

Instalacja zakończyła się dosyć szybko. W tym momencie pozostało tylko skopiować pliki do katalogu /boot, wygenerować nowy ramdisk oraz zaktualizować bootloader.

```
root@localhost:/usr/src/linux-6.9.1# cp arch/x86_64/boot/bzImage /boot/vmlinuz-new-custom-6.9.1
root@localhost:/usr/src/linux-6.9.1# cp System.map /boot/System.map-new-custom-6.9.1
root@localhost:/usr/src/linux-6.9.1# cp .config /boot/config-new-custom-6.9.1
root@localhost:/usr/src/linux-6.9.1# cd /boot
root@localhost:/boot# rm System.map
root@localhost:/boot# ln -s System.map-new-custom-6.9.1 System.map-
root@localhost:/boot# rm System.map-
root@localhost:/boot# ln -s System.map-new-custom-6.9.1 System.map
root@localhost:/boot# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 6.9.1
#
# 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 6.9.1 -f ext4 -r /dev/sda3 -m ext4 -u -o /boot/initrd.gz
root@localhost:/boot# ^C
root@localhost:/boot# mkinitrd -c -k 6.9.1 -f ext4 -r /dev/sda3 -m ext4 -u -o /boot/initrd-new-custom-6.9.1
51307 blocks
/boot/initrd-new-custom-6.9.1 created.
Be sure to run lilo again if you use it.
root@localhost:/boot#
```

```
root@localhost:/# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found linux image: /boot/vmlinuz-new-custom-6.9.1
Found initrd image: /boot/initrd.gz
Found linux image: /boot/vmlinuz-huge-5.15.19
Found initrd image: /boot/initrd.gz
Found linux image: /boot/vmlinuz-huge
Found initrd image: /boot/initrd.gz
Found linux image: /boot/vmlinuz-generic-5.15.19
Found initrd image: /boot/initrd.gz
Found linux image: /boot/vmlinuz-generic
Found initrd image: /boot/initrd.gz
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
Adding boot menu entry for UEFI Firmware Settings ...
done
root@localhost:/#
```

Po restarcie systemu z nowym kernelem:

```
127.0.0.1 - PuTTY
Using username "root".
Keyboard-interactive authentication prompts from server:
| Password:
End of keyboard-interactive prompts from server
Last login: Sun Jun 16 18:16:31 2024 from 10.0.2.2
Linux 6.9.1.
root@localhost:~# uname -r
6.9.1
root@localhost:~#
```

Ponownie zmierzyłem czas startu systemu, wykonując 3 próby.

1) 26.14 s
2) 22.17 s
3) 24.48 s

Średnia: 24.26 s

4. Wnioski i podsumowanie

Moje odczucia po wykonaniu zadania:

Konfiguracja i kompilacja kernela przy użyciu obu metod jest dosyć niewygodna – głównie z powodu ilości istniejących modułów i komponentów. Przejrzenie wszystkiego ręcznie zajęłoby zbyt wiele czasu, nie wspominając o tym że nie wiadomo do czego służy większość z nich. Dlatego potrzebne są narzędzia które poradzą sobie z tym automatycznie. Jednak nawet pomimo ich istnienia łatwo jest popełnić błąd, np. podczas wpisywania polecenia. Na prawdziwym sprzęcie takie błędy są jeszcze bardziej szkodliwe niż na maszynie wirtualnej. Mam wrażenie że cały ten proces (kompilacja, instalacja, bootowanie kernela) powinien zostać w pewien sposób usprawniony

i zaktualizowany, jeśli to możliwe. To prowadzi do wniosku, że mniej zaawansowani użytkownicy systemu Linux nie będą sobie tym zaprzątać głowy. Łatwo jest podchodzić do tego w ten sposób: „jeśli coś działa to tego nie naprawiaj”. W tym przypadku – nie aktualizuj kernela dopóki stara wersja działa dobrze. Zwłaszcza jeśli po aktualizacji system może nie wstać. Tematem warto się zainteresować np. w przypadku odkrycia jakiejś luki bezpieczeństwa w starej wersji jądra, albo w celu zwiększenia wydajności i prędkości działania systemu.

Porównanie metod kompilacji:

Obie metody nie różnią się bardzo pod względem skomplikowania i poziomu trudności. Tak naprawdę ograniczają się do wpisania kilku poleceń. Pewnych wniosków dostarcza spojrzenie na czas kompilacji i bootowania systemu – w przypadku nowej metody kompilacja modułów trwała niecałe 40 minut, a w przypadku starej – lekko ponad minutę. To pokazuje że nowa metoda aktywuje dużo dodatkowych modułów których system może nie potrzebować. Ma to odzwierciedlenie również w czasie rozruchu systemu, który jest prawie dwa razy dłuższy przy użyciu nowej metody. Nie jest to aż tak ważne przy nowszym, szybszym sprzęcie, jednak starsze i mniej wydajne konfiguracje powinny korzystać ze starego rozwiązania. Pozytywna strona nowego podejścia jest taka, że dużo większa ilość modułów i sterowników może np. zwiększyć kompatybilność systemu z urządzeniami zewnętrznymi.