Kompilacja kernela

Mikołaj Cekut

16 czerwca 2022

1 Ogólna konfiguracja

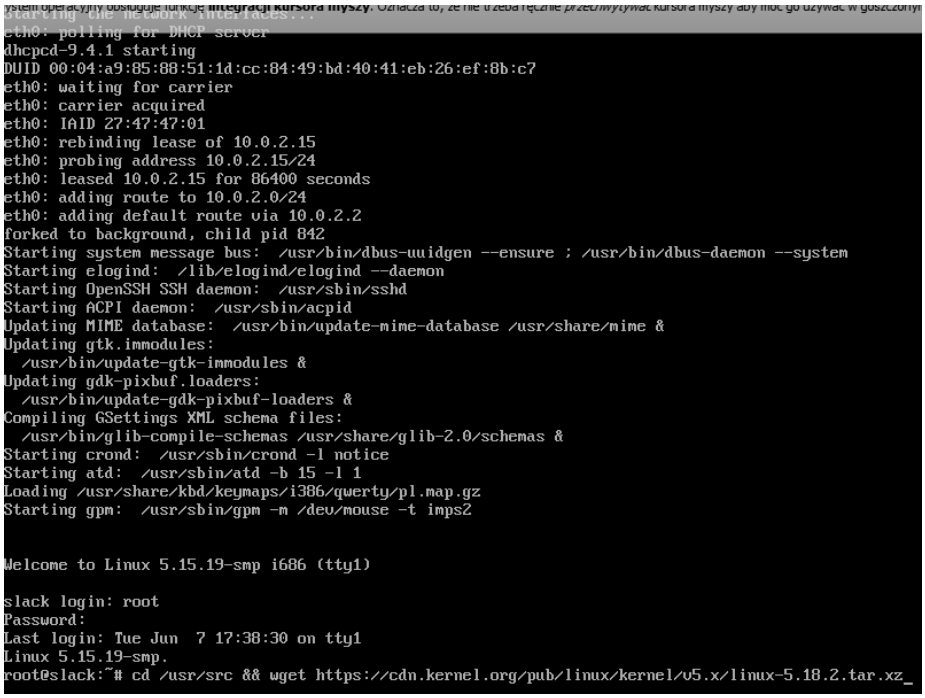
Na początku, niezależnie od metody kompilacji jądra należy pobrać pliki źródłowe wersji jądra, którą chcemy zainstalować. Możemy skorzystać ze strony: https://www.kernel.org/. Wykorzystamy wersję: **5.18.2**. Link do pobrania tej wersji możemy znaleźć pod adresem:

- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.2.tar.xz

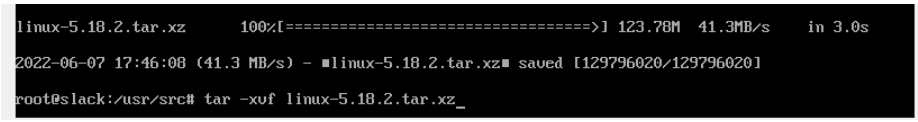
Aby pobrać plik za pomocą konsoli Linuxa korzystamy z polecenia:

- cd /usr/src && wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.2.tar.xz

Dzięki temu poleceniu przeniesiemy się do folderu **/usr/src** i pobierzemy plik. Plik jest skompresowany, więc musimy go rozpakować za pomocą polecenia **tar -xvf**



Rysunek 1: Pobieranie paczki



Rysunek 2: Rozpakowanie paczki

2 Metoda stara

Aktualną konfigurację kernela trzeba przekopiować do pliku **.config** w folderze **/usr/src/linux-5.18.2**

```
linux-5.18.2/  linux-5.18.2.tar.xz
root@slack:/usr/src# cd linux-5.18.2
root@slack:/usr/src/linux-5.18.2# zcat /proc/config.gz > .config
```

Rysunek 3: Kopiowanie starego configa do nowego

Następnie należy użyć polecenia **make localmodconfig** aby wygenerowac plik. Polecenia wyświetliło nam, że korzysta z pliku .config oraz pozwala nam na konfigurację poszczególnych modułów jądra. Używam domyślnej konfiguracji modułów - aby to zrobić, gdy otrzymamy zapytanie o moduł wciskamy **ENTER**.

```
root@slack:/usr/src# cd linux-5.18.2
root@slack:/usr/src/linux-5.18.2# zcat /proc/config.gz > .config
root@slack:/usr/src/linux-5.18.2# ls -l .config
-rw-r--r-- 1 root root 237825 Jun  7 17:50 .config
root@slack:/usr/src/linux-5.18.2# make localmodconfig
```

Rysunek 4: Make localmodconfig

```
  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
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)
```

Rysunek 5: Output make localmodconfig

Kolejnym krokiem jest kompilacja bazowego jądra za pomocą komendy **make -j4 bzImage** oraz modułów jądra za pomocą komendy **make -j4 modules**.

```
root@slack:~# cd /usr/src/linux-5.18.2
root@slack:/usr/src/linux-5.18.2# make -j4 bzImage_
```

Rysunek 6: Make -j4 bzImage

```
  CC      arch/x86/boot/video-bios.o
  HOSTCC  arch/x86/boot/tools/build
  CPUSTR  arch/x86/boot/cpustr.h
  CC      arch/x86/boot/compressed/error.o
  CC      arch/x86/boot/cpu.o
  OBJCOPY arch/x86/boot/compressed/vmlinux.bin
  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-5.18.2# _
```

Rysunek 7: Wynik kompilacji bazowej

```
root@slack:/usr/src/linux-5.18.2# make -j4 modules_
```

Rysunek 8: Make -j4 modules

```
  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]  drivers/virt/vboxguest/vboxguest.ko
  LD [M]  net/802/garp.ko
  LD [M]  net/802/mrp.ko
  LD [M]  net/802/p8022.ko
  LD [M]  net/802/psnap.ko
  LD [M]  net/802/stp.ko
  LD [M]  net/8021q/8021q.ko
  LD [M]  net/ipv6/ipv6.ko
  LD [M]  net/llc/llc.ko
  LD [M]  net/rfkill/rfkill.ko
  LD [M]  net/wireless/cfg80211.ko
  LD [M]  sound/ac97_bus.ko
  LD [M]  sound/core/snd-pcm.ko
  LD [M]  sound/core/snd-timer.ko
  LD [M]  sound/core/snd.ko
  LD [M]  sound/pci/ac97/snd-ac97-codec.ko
  LD [M]  sound/pci/snd-intel8x0.ko
  LD [M]  sound/soundcore.ko
root@slack:/usr/src/linux-5.18.2# _
```

Rysunek 9: Wynik kompilacji modułów

Po skompilowaniu modułów należy je zainstalować komendą **make modules_install**.

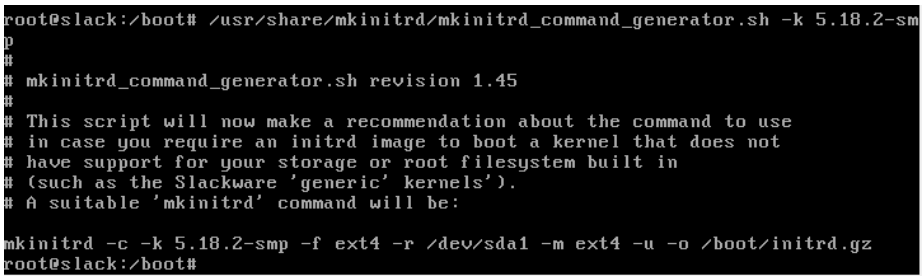
```
  INSTALL /lib/modules/5.18.2-smp/kernel/drivers/powercap/intel_rapl_msr.ko
  INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/fb_sys_fops.ko
  INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/syscopyarea.ko
  INSTALL /lib/modules/5.18.2-smp/kernel/drivers/video/fbdev/core/sysfillrect.ko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

Rysunek 10: Wynik instalacji modułów

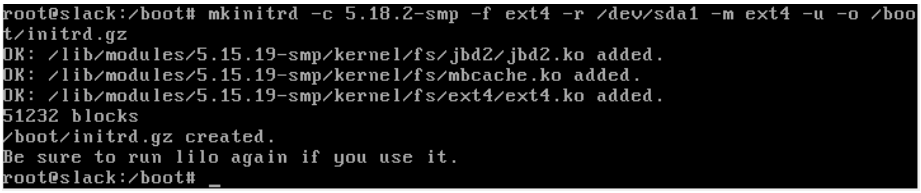
Kolejnym krokiem jest skopiowanie pozostałych plików potrzebnych do uruchomienia jądra.

Rysunek 11: Kopiowanie potrzebnych plików do katalogu /boot

Po zainstalowaniu modułów i skopiowania plików następnym krokiem jest wygenerowaniu ramdisku. Za pomocą komendy **mkinitrd_command_generator.sh -k 5.18.2-smp** otrzymamy polecenie, które musimy wykorzystać do wygenerowania ramdisku.

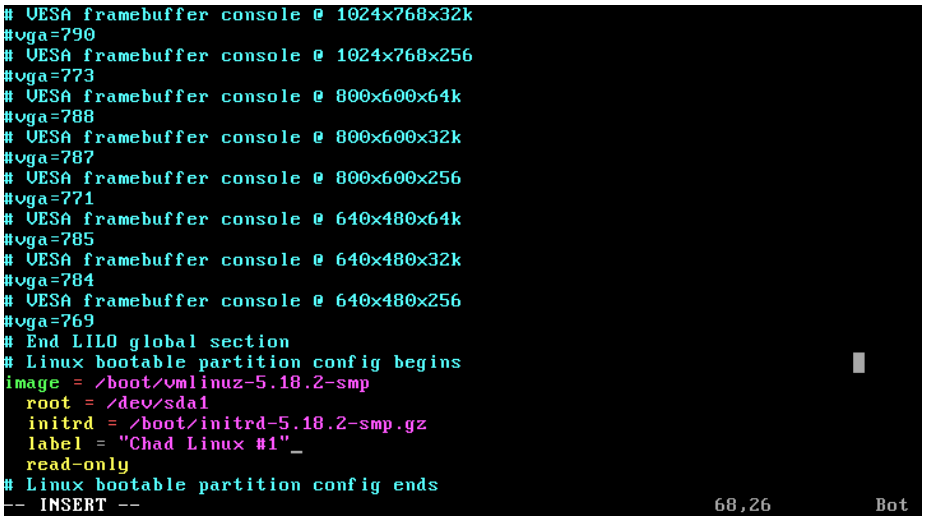


Rysunek 12: mkinitrd_command_generator.sh -k 5.18.2-smp

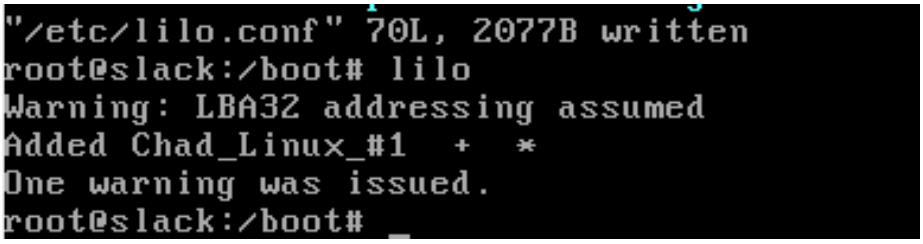


Rysunek 13: Użycie wygenerowanej komendy

Ostatnim krokiem jaki został przed możliwością uruchomenia nowego kernela jest dodanie wpisu o nim do lilo.conf oraz użycie komendy **lilo**



Rysunek 14: Konfiguracja w pliku lilo.conf



Rysunek 15: Wywołanie komendy lilo

Teraz możemy uruchomić maszynę ponownie. Jak widać nadpisałem poprzednią konfigurację kernela swoją własną, nową, tak więc widać tylko jeden wpis - nowo wgranego kernela.

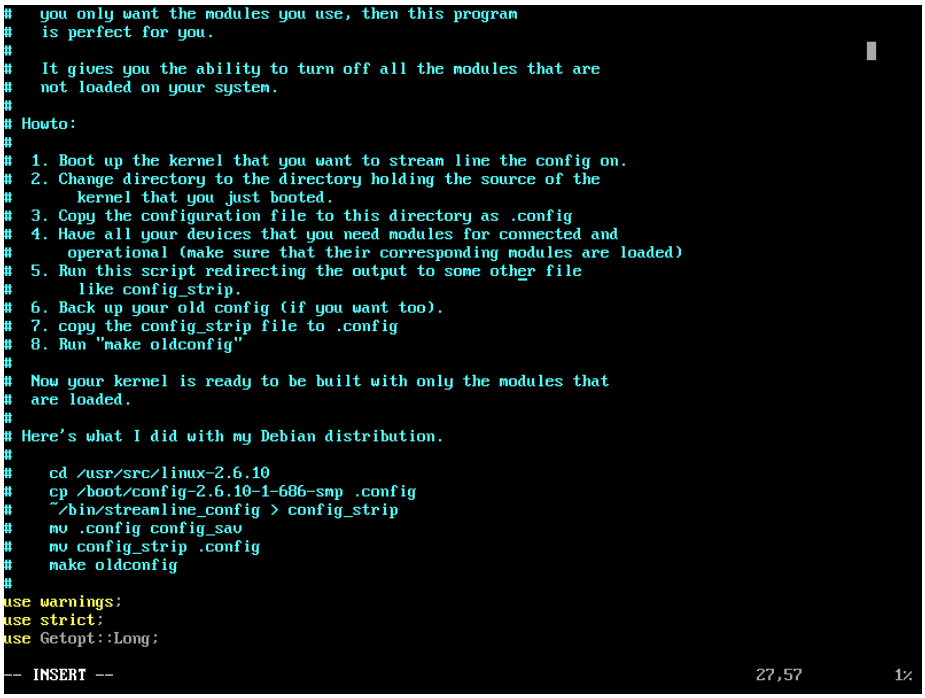
Rysunek 16: Ekran startowy

Uruchamiamy więc nasz kernel. Jak widać udało nam się zalogować do systemu, tak więc kompilacja okazała się sukcesem, jednak pragnę zauważyć, iż system uruchamiał się bardzo długo (około 3-4 minut).

Rysunek 17: Logowanie udane

3 Metoda nowa

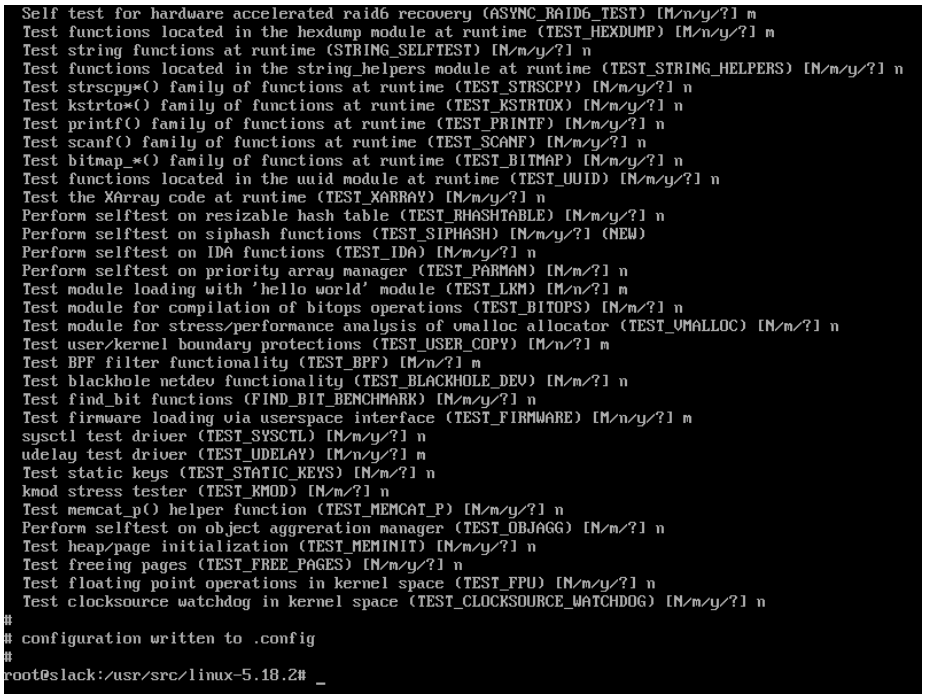
Korzystając z metody nowej wykorzystujemy skrypt **streamline_config.pl** zlokalizowany w katalogu *./scripts/kconfig/*.



Rysunek 18: Skrypt streamline_config.pl otwarty za pomocą vim zawiera w sobie tutorial

Rysunek 19: Wykorzystanie skryptu

Następnie używamy komendy **make oldconfig** aby stworzyć konfigurację.



Rysunek 20: make oldconfig

Następne etapy (make bzImage, make modules itd.) zostały opisane w poprzedniej metodzie, cały ten proces jest dokładnie taki sam jak poprzednio, więc przejdziemy do wyniku kompilacji. Jak widać na ekranie startowym pojawił się nowo skompilowany kernel. Niestety system nie chce się uruchomić, pojawia się błąd, po czym maszyna się źamraża". Po próbie uruchomienia ponownie maszyny problem się powtarza.

Rysunek 21: Ekran startowy po wykonaniu drugiej metody

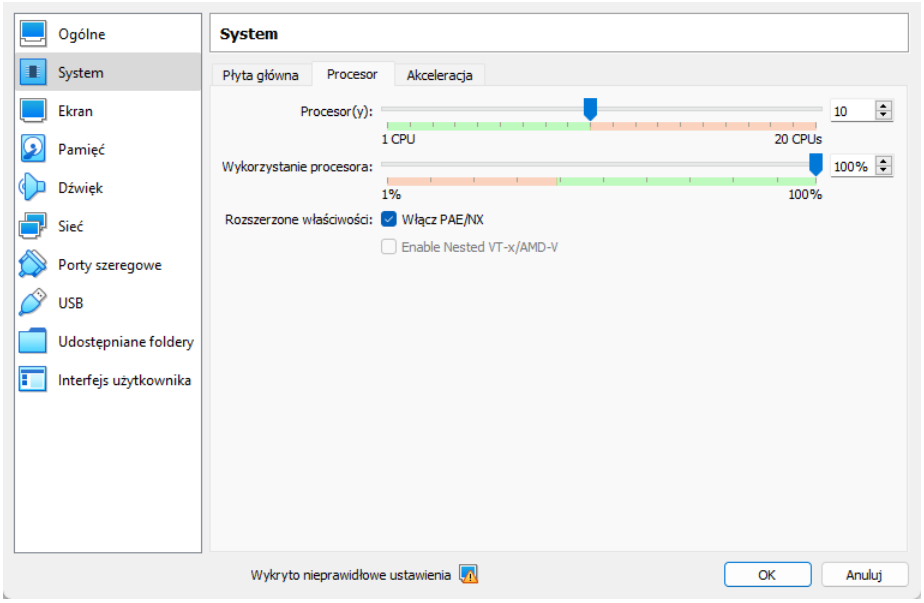
```
 GiB)
[  279.156634] sd 0:0:0:0: [sda] Write Protect is off
[  279.193922] sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, does
n't support DPO or FUA
[  280.408761]  sda: sda1 sda2 sda3
[  281.162305] sd 0:0:0:0: [sda] Attached SCSI disk
[  281.546770] Freeing unused kernel image (initmem) memory: 1048K
[  281.954707] Write protecting kernel text and read-only data: 22136k
[  282.032606] NX-protecting the kernel data: 10148k
[  282.129952] rodata_test: all tests were successful
[  282.167115] Run /init as init process
[  308.285178] clocksource: timekeeping watchdog on CPU6: Marking clocksource 't
sc-early' as unstable because the skew is too large:
[  308.373090] clocksource:                       'acpi_pm' wd_nsec: 2075442269
wd_now: 78c476 wd_last: 76863 mask: ffffff
[  308.460197] clocksource:                       'tsc-early' cs_nsec: 168179036
074 cs_now: cfe59b74c cs_last: 9a86f4010 mask: ffffffffffffffff
[  308.570698] clocksource:                       'tsc-early' is current clockso
urce.
[  308.646888] tsc: Marking TSC unstable due to clocksource watchdog
[  308.697957] TSC found unstable after boot, most likely due to broken BIOS. Us
e 'tsc=unstable'.
[  308.778063] sched_clock: Marking unstable (308623927644, 74024465)<-(31688471
0808, -8186758354)
_
```

Rysunek 22: Błąd uruchomienia drugiej metody

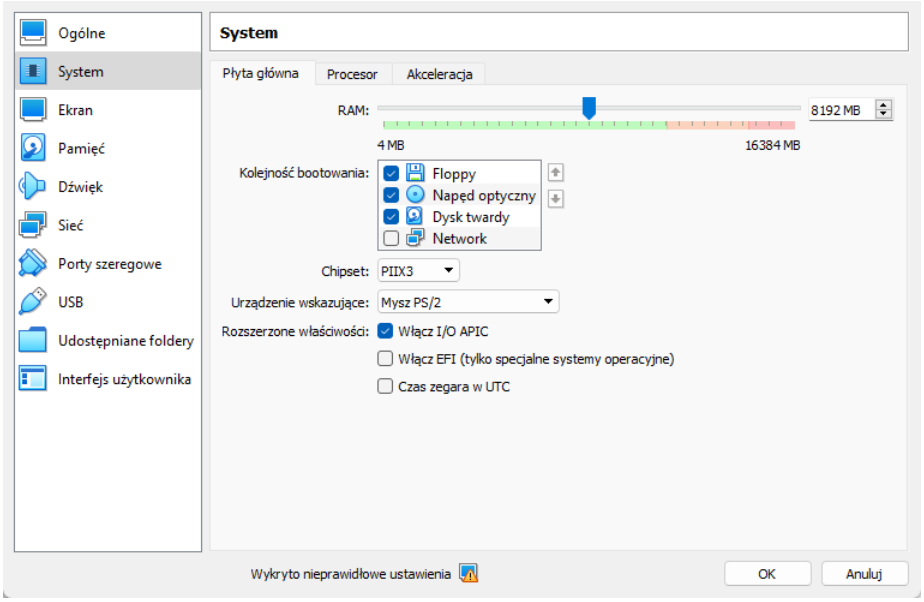
4 Podsumowanie

Porównując obie metody, co do samego procesu tworzenia nie ma większych różnić, może oprócz tego, że nową metodą moduły kompilują się wolniej. Jednak co do wyniku kompilacji nie miałem możliwości ich porównać, ponieważ po kompilacji metodą starą linux niestety nie uruchomił się.

Dodam jeszcze, że obie konfiguracje wykonywałem korzystając z tej konfiguracji maszyny wirtualnej.



Rysunek 23: Konfiguracja maszyny #1



Rysunek 24: Konfiguracja maszyny #2