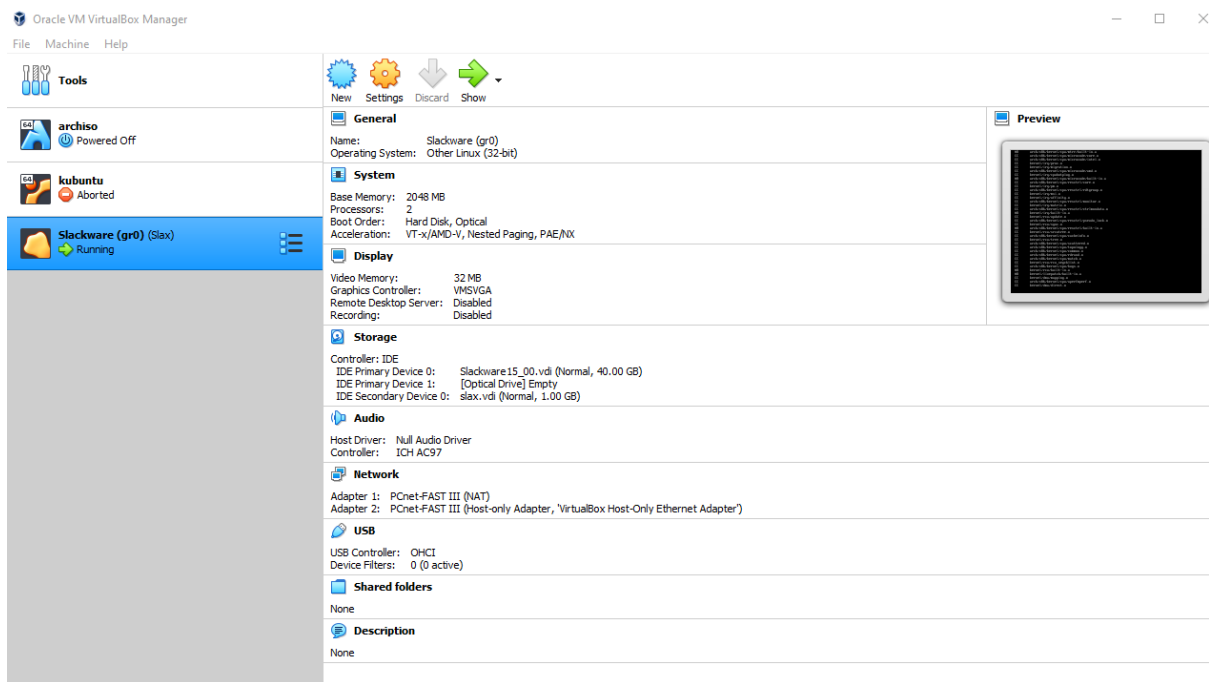


# Kompilacja Kernela Linux

Metoda Stara i Nowa

Igor Mazur, 290989

## 1. Przygotowanie



Pracę rozpocząłem od sprawdzeniu ustawień wirtualnej maszyny przed jej odpaleniem, jak widać używam 2 processory i 2 GB RAM.

Po włączeniu i zalogowaniu się do systemu jako użytkownik **root** z hasłem **slack2022#**, wywołuję komendę **neofetch** w celach pokazania informacji o systemie.



```
Slackware (gr0) (Slax) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

root@slack:~#
root@slack:~#
root@slack:~#
root@slack:~# neofetch

          :
          :
          :
          :llcccccllllllll:
          :lc          dc:
          :cl  clllcclll  oc:
          :o  lc:ccccco  oc:
          :o  cccclc:clcc:
          :lc      cclccclc:
          :lccclcc      lc:
          :ccclcc:ccclcc  oc:
          :o  l:lllllll  lc:
          :cllo  clcllcccll  o:
          :occllo      clc:
          :ocl:ccslclccclclccclclc:
          :ocllccccccccccccclllllllllll:
          :lcc1lccccccccccccccccccccco:
          :
          :
          :
          :

root@slack:~# cd /usr/src
root@slack:/usr/src# wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.tar.xz
--2022-07-01 19:09:22-- https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.18.tar.xz
Translacja cdn.kernel.org (cdn.kernel.org)... 151.101.113.176, 2a04:4e42:3::432
■cznienie si■ z cdn.kernel.org (cdn.kernel.org)|151.101.113.176|:443... po■czono.
■danie HTTP wys■ano, oczekiwanie na odpowied■... 200 OK
Dugo■: 129790264 (124M) [application/x-xz]
Zapis do: `linux-5.18.tar.xz'

linux-5.18.tar.xz      18%[====>                ] 22,58M  2,42MB/s  eta 35s
```

Figure 2 Pobranie wget jądra

Po pobraniu jądra linuxa, musimy je teraz rozpakować używając np. komendy:

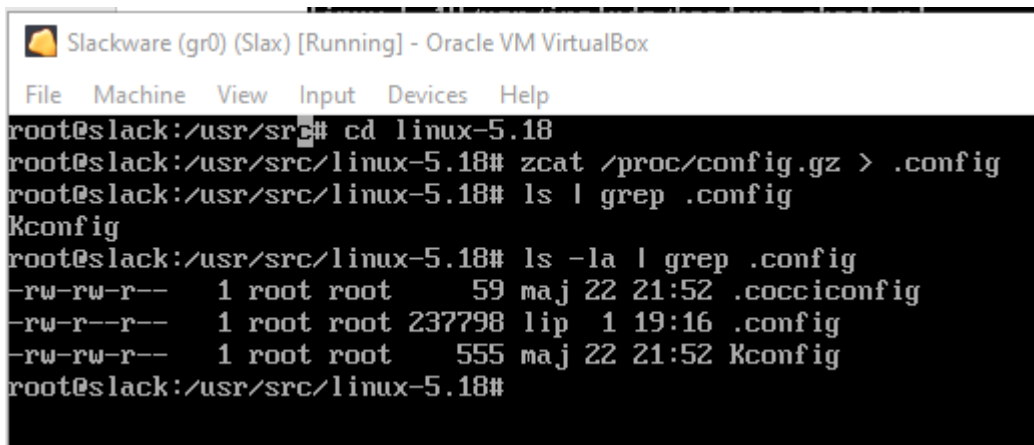
***tar -xvf linux-5.18.tar.xz***

```
linux-5.18/usr/default_cpio_list
linux-5.18/usr/gen_init_cpio.c
linux-5.18/usr/gen_initramfs.sh
linux-5.18/usr/include/
linux-5.18/usr/include/.gitignore
linux-5.18/usr/include/Makefile
linux-5.18/usr/include/headers_check.pl
linux-5.18/usr/initramfs_data.S
linux-5.18/virt/
linux-5.18/virt/Makefile
linux-5.18/virt/kvm/
linux-5.18/virt/kvm/Kconfig
linux-5.18/virt/kvm/Makefile.kvm
linux-5.18/virt/kvm/async_pf.c
linux-5.18/virt/kvm/async_pf.h
linux-5.18/virt/kvm/binary_stats.c
linux-5.18/virt/kvm/coalesced_mmio.c
linux-5.18/virt/kvm/coalesced_mmio.h
linux-5.18/virt/kvm/dirty_ring.c
linux-5.18/virt/kvm/eventfd.c
linux-5.18/virt/kvm/irqchip.c
linux-5.18/virt/kvm/kvm_main.c
linux-5.18/virt/kvm/kvm_mm.h
linux-5.18/virt/kvm/pfncache.c
linux-5.18/virt/kvm/vfio.c
linux-5.18/virt/kvm/vfio.h
linux-5.18/virt/lib/
linux-5.18/virt/lib/Kconfig
linux-5.18/virt/lib/Makefile
linux-5.18/virt/lib/irqbypass.c
root@slack:/usr/src#
root@slack:/usr/src#
root@slack:/usr/src#
root@slack:/usr/src#
root@slack:/usr/src#
root@slack:/usr/src#
root@slack:/usr/src#
```

Figure 3 Wypakowanie tar -xvf linux-5.18.tar.xz

## Metoda stara

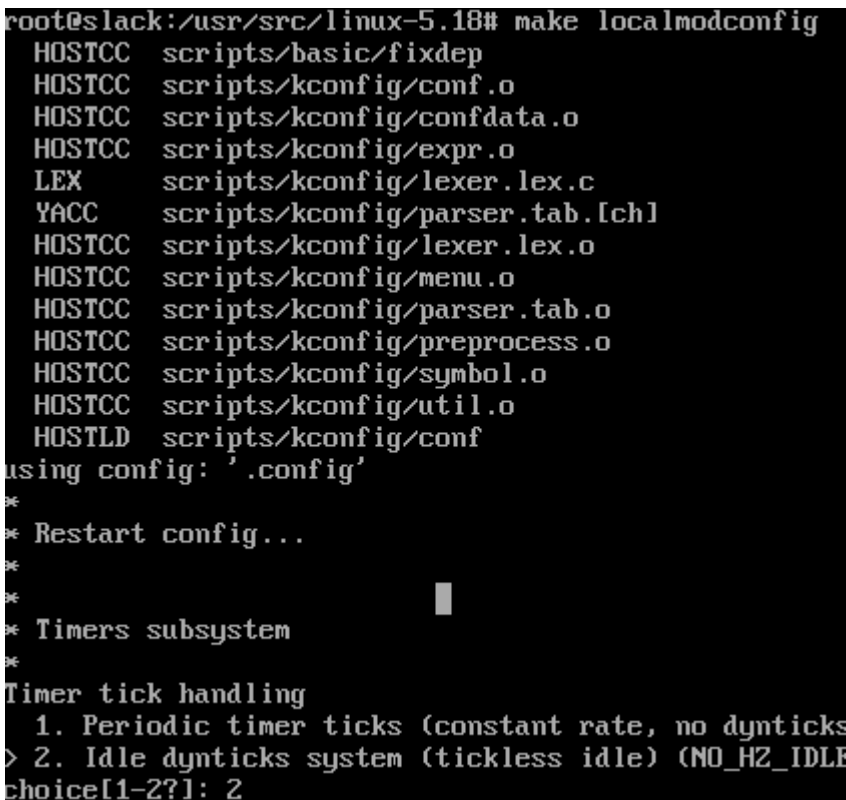
Na początku tworze kopie aktualnej konfiuracji kernel do pliku .config



```
Slackware (gr0) (Slax) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@slack:/usr/src# cd linux-5.18
root@slack:/usr/src/linux-5.18# zcat /proc/config.gz > .config
root@slack:/usr/src/linux-5.18# ls | grep .config
Kconfig
root@slack:/usr/src/linux-5.18# ls -la | grep .config
-rw-rw-r-- 1 root root 59 maj 22 21:52 .cocciconfig
-rw-r--r-- 1 root root 237798 lip 1 19:16 .config
-rw-rw-r-- 1 root root 555 maj 22 21:52 Kconfig
root@slack:/usr/src/linux-5.18#
```

Figure 4 Wykonanie kopii starej konfiguracji

Jak już utworzy się plik config, kolejnym krokiem będzie wykonanie komendy **make localmodconfig**. Jest tutaj bardzo dużo opcji do przeglądu, ale bezpiecznie możemy je wszystkie zostawić na domyślnych, czyli wciskając bardzo dużo okropnych enterów.



```
root@slack:/usr/src/linux-5.18# 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)
  > 2. Idle dynticks system (tickless idle) (NO_HZ_IDLE)
choice[1-2?]: 2
```

Wynikiem tej komendy będzie to:

```

Test functions located in the hexdump module at runtime (TEST_HEXDUMP) [N/m/y/?] n
Test string functions at runtime (STRING_SELFTEST) [N/m/y/?] n
Test functions located in the string_helpers module at runtime (TEST_STRING_HELPERS) [N/m/y/?] n
Test strscpy*() family of functions at runtime (TEST_STRSCPY) [N/m/y/?] n
Test kstrt*() family of functions at runtime (TEST_KSTRTX) [N/m/y/?] n
Test printf() family of functions at runtime (TEST_PRINTF) [N/m/y/?] n
Test scanf() family of functions at runtime (TEST_SCANF) [N/m/y/?] n
Test bitmap_*() family of functions at runtime (TEST_BITMAP) [N/m/y/?] n
Test functions located in the uuid module at runtime (TEST_UUID) [N/m/y/?] n
Test the XArray code at runtime (TEST_XARRAY) [N/m/y/?] n
Perform selftest on resizable hash table (TEST_RHASHTABLE) [N/m/y/?] n
Perform selftest on siphash functions (TEST_SIPHASH) [N/m/y/?] (NEW)
Perform selftest on IDA functions (TEST_IDA) [N/m/y/?] n
Test module loading with 'hello world' module (TEST_LKM) [N/m/?] n
Test module for compilation of bitops operations (TEST_BITOPS) [N/m/?] n
Test module for stress/performance analysis of umalloc allocator (TEST_UMALLOC) [N/m/?] n
Test user/kernel boundary protections (TEST_USER_COPY) [N/m/?] n
Test BPF filter functionality (TEST_BPF) [N/m/?] n
Test blackhole netdev functionality (TEST_BLACKHOLE_DEV) [N/m/?] n
Test find_bit functions (FIND_BIT_BENCHMARK) [N/m/y/?] n
Test firmware loading via userspace interface (TEST_FIRMWARE) [N/m/y/?] n
sysctl test driver (TEST_SYSCTL) [N/m/y/?] n
udelay test driver (TEST_UDELAY) [N/m/y/?] n
Test static keys (TEST_STATIC_KEYS) [N/m/?] n
kmod stress tester (TEST_KMOD) [N/m/?] n
Test memcat_p() helper function (TEST_MEMCAT_P) [N/m/y/?] n
Test heap/page initialization (TEST_MEMINIT) [N/m/y/?] n
Test freeing pages (TEST_FREE_PAGES) [N/m/y/?] n
Test floating point operations in kernel space (TEST_FPU) [N/m/y/?] n
Test clocksource watchdog in kernel space (TEST_CLOCKSOURCE_WATCHDOG) [N/m/y/?] n
}
# configuration written to .config
}
root@slack:/usr/src/linux-5.18#
root@slack:/usr/src/linux-5.18#

```

Figure 5 make localmodconfig

W tym momencie jesteśmy gotowi już do rozpoczęcia kompilacji jądra. Ten proces zaczynam poprzez:

***make -j2 bzImage***

```

root@slack:/usr/src/linux-5.18# make -j2 bzImage
SYSHDR arch/x86/include/generated/uapi/asm/unistd_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
SYSHDR arch/x86/include/generated/uapi/asm/unistd_64.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
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_x32.h
SYSTBL arch/x86/include/generated/asm/syscalls_32.h
HOSTCC arch/x86/tools/relocs_32.o
UPD include/config/kernel.release
WRAP arch/x86/include/generated/asm/early_ioremap.h
HOSTCC arch/x86/tools/relocs_64.o
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/mmiowb.h
WRAP arch/x86/include/generated/asm/module.lds.h

```

Figure 6 Kompilacja na dwóch rdzeniach

```

LD arch/x86/boot/setup.coff
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# _

```

Po kompilacji jądra, kompilujemy moduły używając:

***make modules***  
***make modules\_install***

```

LD [M] sound/pci/snd-intel8x0.ko
CC [M] sound/soundcore.mod.o
LD [M] sound/soundcore.ko
root@slack:/usr/src/linux-5.18#

```

Figure 7 Kompilacja modułów

```

INSTALL /lib/modules/5.18.0-smp/kernel/sound/core/snd-timer.ko
INSTALL /lib/modules/5.18.0-smp/kernel/sound/core/snd.ko
INSTALL /lib/modules/5.18.0-smp/kernel/sound/pci/ac97/snd-ac97-codec.ko
INSTALL /lib/modules/5.18.0-smp/kernel/sound/pci/snd-intel8x0.ko
INSTALL /lib/modules/5.18.0-smp/kernel/sound/soundcore.ko
DEPMOD /lib/modules/5.18.0-smp
root@slack:/usr/src/linux-5.18# _

```

Figure 8 make modules\_install

Potrzebujemy skopiować pliki potrzebne do uruchomienia nowego jądra do katalogu /boot:

```

cp arch/x86/boot/bzImage /boot/vmlinuz-starametoda-5.18-smp
cp System.map /boot/System.map-starametoda-5.18-smp
cp .config /boot/config-starametoda-5.18-smp

```

```

root@slack:/usr/src/linux-5.18# cp arch/x86/boot/bzImage /boot/vmlinuz-starametoda-5.18-smp
root@slack:/usr/src/linux-5.18# cp System.map /boot/System.map-starametoda-5.18-smp
root@slack:/usr/src/linux-5.18# cp .config /boot/config-starametoda-5.18-smp
root@slack:/usr/src/linux-5.18# _

```

Figure 9 Kopiowanie plików nowego jądra

Kolejnym krokiem jest zlinkowanie pliku System.map

```

root@slack:/usr/src/linux-5.18# cd /boot
root@slack:/boot# ls -la | grep System.map
lrwxrwxrwx 1 root root 31 kwi 25 19:06 System.map -> System.map-huge-smp-5.15.27-smp
-rw-r--r-- 1 root root 3883385 mar 9 02:44 System.map-generic-5.15.27
-rw-r--r-- 1 root root 4020483 mar 9 04:00 System.map-generic-smp-5.15.27-smp
-rw-r--r-- 1 root root 5333639 mar 9 02:41 System.map-huge-5.15.27
-rw-r--r-- 1 root root 5473713 mar 9 03:55 System.map-huge-smp-5.15.27-smp
-rw-r--r-- 1 root root 5453466 lip 1 20:22 System.map-starametoda-5.18-smp
root@slack:/boot# rm System.map
root@slack:/boot# ln -s System.map-starametoda-5.18-smp System.map
root@slack:/boot#

```

Figure 10 System.map

```

root@slack:/boot# cd /usr/src/linux-5.18
root@slack:/usr/src/linux-5.18# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.0-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 5.18.0-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
root@slack:/usr/src/linux-5.18# _

mkinitrd -c -k 5.18.0-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
root@slack:/usr/src/linux-5.18# mkinitrd -c -k 5.18.0-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/i
nitrd.gz
49039 bloków
/boot/initrd.gz created.
Be sure to run lilo again if you use it.
root@slack:/usr/src/linux-5.18# _

```

Figure 11 Generowanie ramdisk

```
mkinitrd -c -k 5.18.0-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
```



```

# End LILO global section
# Linux bootable partition config begins
image = /boot/vmlinuz
  root = /dev/sda1
  label = "Slackware 15.0"
  read-only
# Linux bootable partition config ends
root@slack:/usr/src/linux-5.18# _

```

Figure 12 Plik lilo.conf

Edytujemy plik aby móc uruchomić nowego kernela, używamy komendy lilo:

*ls /boot*  
*lilo*

```

root@slack:/usr/src/linux-5.18# ls /boot
README.initrd@      config-huge-smp-5.15.27-smp  tuxlogo.bmp
System.map@         config-starametoda-5.18-smp  tuxlogo.dat
System.map-generic-5.15.27  elilo-ia32.efi*             vmlinuz@
System.map-generic-smp-5.15.27-smp  elilo-x86_64.efi*          vmlinuz-generic@
System.map-huge-5.15.27      grub/                        vmlinuz-generic-5.15.27
System.map-huge-smp-5.15.27-smp  initrd-tree/               vmlinuz-generic-smp@
System.map-starametoda-5.18-smp  initrd.gz                  vmlinuz-generic-smp-5.15.27-smp
boot.0800               inside.bmp                  vmlinuz-huge@
boot_message.txt        inside.dat                  vmlinuz-huge-5.15.27
config@                 map                          vmlinuz-huge-smp@
config-generic-5.15.27   onlyblue.bmp               vmlinuz-huge-smp-5.15.27-smp
config-generic-smp-5.15.27-smp  onlyblue.dat              vmlinuz-starametoda-5.18-smp
config-huge-5.15.27      slack.bmp
root@slack:/usr/src/linux-5.18# lilo
Warning: LBA32 addressing assumed
Warning: Unable to determine video adapter in use in the present system.
Warning: Video adapter does not support VESA BIOS extensions needed for
display of 256 colors. Boot loader will fall back to TEXT only operation.
Added Slackware_15.0 *
3 warnings were issued.
root@slack:/usr/src/linux-5.18#

```

Figure 13 Uruchomienie lilo

Wszystko już powinno być gotowe do ponownego uruchomienia maszyny. Mam trzy warningi co do wirtualnej maszyny.

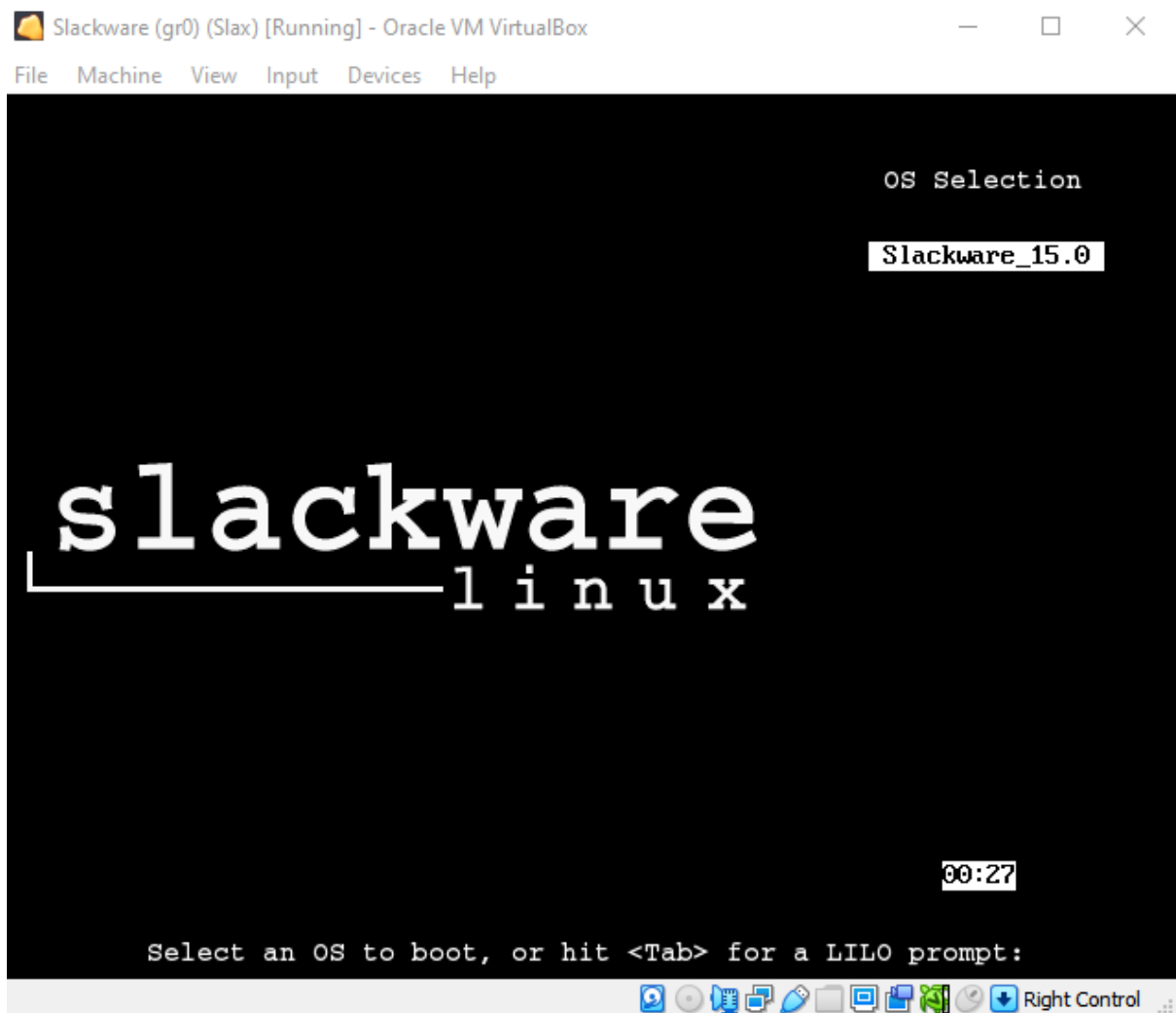
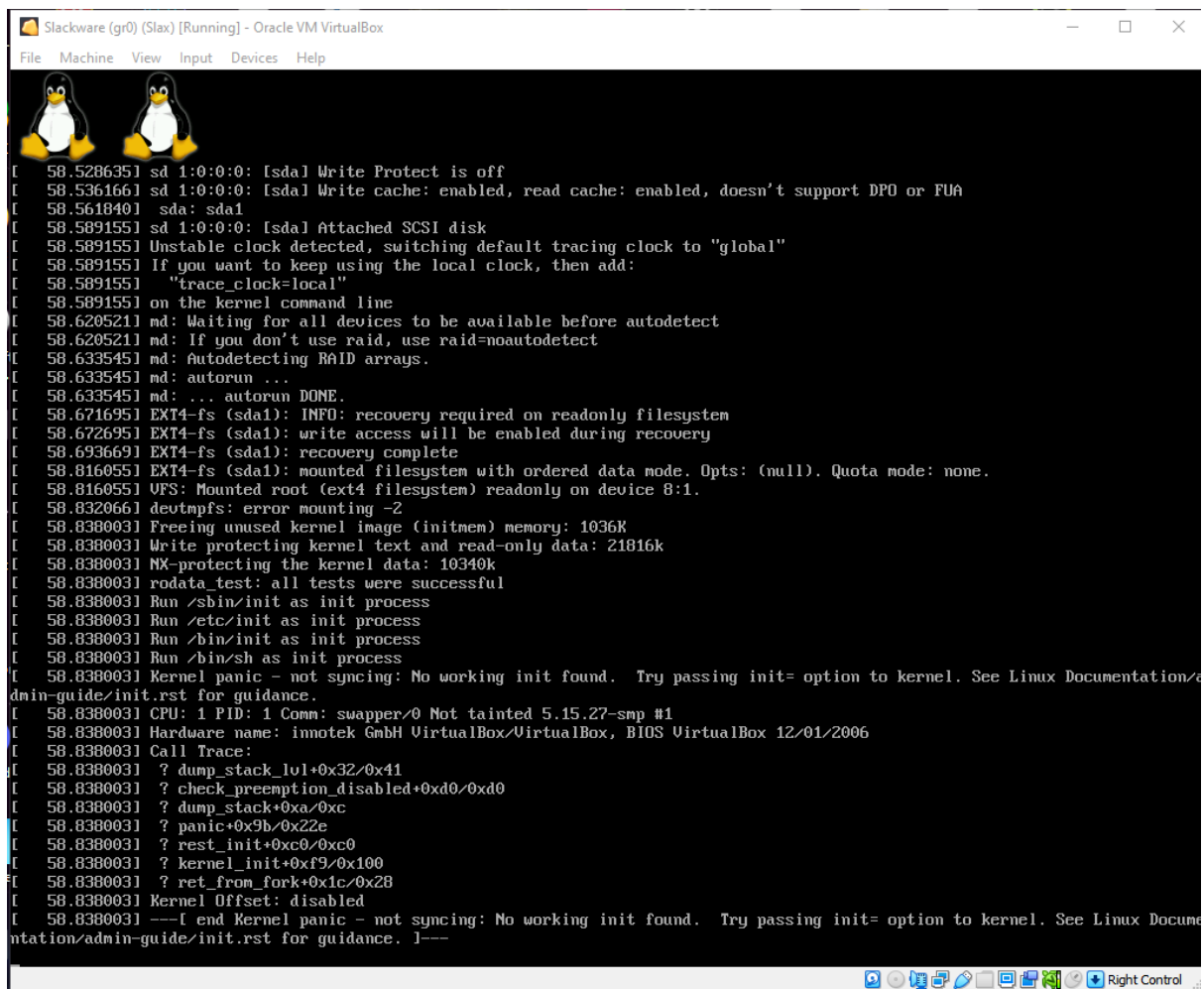


Figure 14 Ponowne uruchomienie



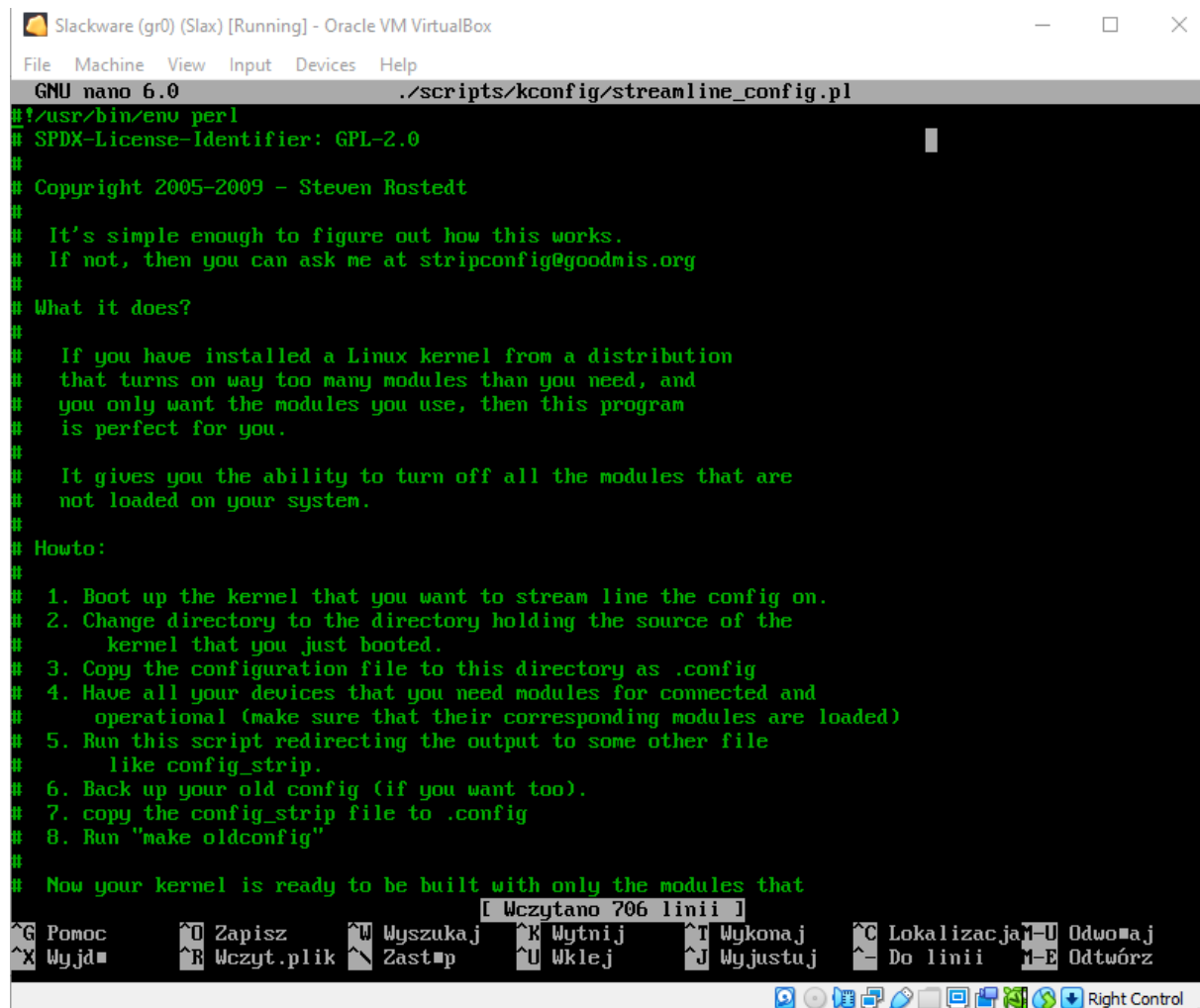
```
Slackware (gr0) (Slax) [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

[ 58.528635] sd 1:0:0:0: [sda] Write Protect is off
[ 58.536166] sd 1:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
[ 58.561840] sda: sda1
[ 58.589155] sd 1:0:0:0: [sda] Attached SCSI disk
[ 58.589155] Unstable clock detected, switching default tracing clock to "global"
[ 58.589155] If you want to keep using the local clock, then add:
[ 58.589155] "trace_clock=local"
[ 58.589155] on the kernel command line
[ 58.620521] md: Waiting for all devices to be available before autodetect
[ 58.620521] md: If you don't use raid, use raid=noautodetect
[ 58.633545] md: Autodetecting RAID arrays.
[ 58.633545] md: autorun ...
[ 58.633545] md: ... autorun DONE.
[ 58.671695] EXT4-fs (sda1): INFO: recovery required on readonly filesystem
[ 58.672695] EXT4-fs (sda1): write access will be enabled during recovery
[ 58.693669] EXT4-fs (sda1): recovery complete
[ 58.816055] EXT4-fs (sda1): mounted filesystem with ordered data mode. Opts: (null). Quota mode: none.
[ 58.816055] VFS: Mounted root (ext4 filesystem) readonly on device 8:1.
[ 58.832066] devtmpfs: error mounting -2
[ 58.838003] Freeing unused kernel image (initmem) memory: 1036K
[ 58.838003] Write protecting kernel text and read-only data: 21816k
[ 58.838003] NX-protecting the kernel data: 10340k
[ 58.838003] rodata test: all tests were successful
[ 58.838003] Run /sbin/init as init process
[ 58.838003] Run /etc/init as init process
[ 58.838003] Run /bin/init as init process
[ 58.838003] Run /bin/sh as init process
[ 58.838003] Kernel panic - not syncing: No working init found. Try passing init= option to kernel. See Linux Documentation/
admin-guide/init.rst for guidance.
[ 58.838003] CPU: 1 PID: 1 Comm: swapper/0 Not tainted 5.15.27-smp #1
[ 58.838003] Hardware name: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
[ 58.838003] Call Trace:
[ 58.838003] ? dump_stack_lvl+0x32/0x41
[ 58.838003] ? check_preemption_disabled+0xd0/0xd0
[ 58.838003] ? dump_stack+0xa/0xc
[ 58.838003] ? panic+0x9b/0x22e
[ 58.838003] ? rest_init+0xc0/0xc0
[ 58.838003] ? kernel_init+0xf9/0x100
[ 58.838003] ? ret_from_fork+0x1c/0x28
[ 58.838003] Kernel Offset: disabled
[ 58.838003] ---[ end Kernel panic - not syncing: No working init found. Try passing init= option to kernel. See Linux Docume
ntation/admin-guide/init.rst for guidance. ]---
```

Figure 15 Sprawdzenie działania nowego jądra

## Metoda nowa

Po ponownym rozpakowaniu wirtualnej maszyny aby móc zacząć od takiego samego stanu początkowego, używamy skryptu `streamline_config.pl` z `/scripts/kconfig/`



The screenshot shows a terminal window titled "Slackware (gr0) (Slax) [Running] - Oracle VM VirtualBox". The terminal is running the GNU nano 6.0 editor, editing the file `./scripts/kconfig/streamline_config.pl`. The script content is as follows:

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

The terminal window also shows a status bar at the bottom with various icons and a "Right Control" button.

Figure 16 Zawartość `streamline_config.pl`

Postępując zgodnie z instrukcjami z pliku `streamline_config.pl`, przechodzimy dalej zgodnie z instrukcjami.

Kopiujemy config a później go uruchomiamy oraz finalnie podmieniamy.

```
root@slack:/usr/src/linux-5.18# cp /boot/config .config
root@slack:/usr/src/linux-5.18# ./scripts/kconfig/streamline_config.pl > config_strip
using config: '.config'
root@slack:/usr/src/linux-5.18# mv .config config.bak
root@slack:/usr/src/linux-5.18# mv config_strip .config
root@slack:/usr/src/linux-5.18# _
```

Figure 17 Początkowe kroki nowej metody

Zgodnie z instrukcją, kolejnym krokiem jest użycie **make oldconfig**

```
root@slack:/usr/src/linux-5.18# 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]
```

Figure 18 make oldconfig

Tak jak i w starej metodzie, zostawiamy wszystkie ustawienia domyślnie, zatwierdzając enterem.

```
Test module for compilation of bitops operations (TEST_BITOPS) [N/m/?] n
Test module for stress/performance analysis of umalloc allocator (TEST_UMALLOC) [N/m/?] n
Test user/kernel boundary protections (TEST_USER_COPY) [N/m/?] n
Test BPF filter functionality (TEST_BPF) [N/m/?] n
Test blackhole netdev functionality (TEST_BLACKHOLE_DEV) [N/m/?] n
Test find_bit functions (FIND_BIT_BENCHMARK) [N/m/y/?] n
Test firmware loading via userspace interface (TEST_FIRMWARE) [N/m/y/?] n
sysctl test driver (TEST_SYSCTL) [N/m/y/?] n
udelay test driver (TEST_UDELAY) [N/m/y/?] n
Test static keys (TEST_STATIC_KEYS) [N/m/?] n
kmod stress tester (TEST_KMOD) [N/m/?] n
Test memcat_p() helper function (TEST_MEMCAT_P) [N/m/y/?] n
Test heap/page initialization (TEST_MEMINIT) [N/m/y/?] n
Test freeing pages (TEST_FREE_PAGES) [N/m/y/?] n
Test floating point operations in kernel space (TEST_FPU) [N/m/y/?] n
Test clocksource watchdog in kernel space (TEST_CLOCKSOURCE_WATCHDOG) [N/m/y/?] n
#
# configuration written to .config
#
root@slack:/usr/src/linux-5.18#
```

Figure 19 Domyślne ustawienia

Ponownie kompilujemy jądro używając **make -j2 bzImage**

```
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# _
```

Figure 20 Wynik kompilacji

Kolejnym krokiem jest kompilacja modułów **make -j2 modules**

```
root@slack:/usr/src/linux-5.18# make -j2 modules
CALL      scripts/atomic/check-atomics.sh
CALL      scripts/checksyscalls.sh

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# _
```

Figure 21 Wynik kompilacji modułów

Teraz jesteśmy gotowni zainstalować moduły używając **make modules\_install**

```
LD [M]    sound/soundcore.ko
root@slack:/usr/src/linux-5.18# make modules_install
INSTALL /lib/modules/5.18.0-smp/kernel/arch/x86/events/intel/intel-cstate.ko
INSTALL /lib/modules/5.18.0-smp/kernel/arch/x86/events/rapl.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/acpi/ac.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/acpi/battery.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/acpi/button.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/acpi/video.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/block/loop.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/char/agp/agpgart.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/char/agp/intel-agp.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/char/agp/intel-gtt.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/gpu/drm/drm.ko
INSTALL /lib/modules/5.18.0-smp/kernel/drivers/gpu/drm/drm_kms_helper.ko
INSTALL /lib/modules/5.18.0-smp/kernel/sound/core/snd.ko
INSTALL /lib/modules/5.18.0-smp/kernel/sound/pci/ac97/snd-ac97-co
INSTALL /lib/modules/5.18.0-smp/kernel/sound/pci/snd-intel8x0.ko
INSTALL /lib/modules/5.18.0-smp/kernel/sound/soundcore.ko
DEPMOD /lib/modules/5.18.0-smp
root@slack:/usr/src/linux-5.18#
```

Figure 22 Wynik instalacji modułów

Następnie robimy tak samo jak w starej metodzie. Kopiujemy pliki do /boot i linkujemy System.map

```
root@slack:/usr/src/linux-5.18# cp arch/x86/boot/bzImage /boot/vmlinuz-nowametoda-5.18-smp
root@slack:/usr/src/linux-5.18# cp System.map /boot/System.map-nowametoda-5.18-smp
root@slack:/usr/src/linux-5.18# cp .config /boot/config-nowametoda-5.18-smp
root@slack:/usr/src/linux-5.18# rm /boot/System.map
root@slack:/usr/src/linux-5.18# cd /boot
root@slack:/boot# ln -s System.map-nowametoda-5.18-smp System.map
root@slack:/boot#
```

Figure 23 Kopiowanie plików

Identycznie jak w starej wersji, generujemy komendę

```

root@slack:/usr/src/linux-5.18# /usr/share/mkinitrd/mkinitrd_command_generator.sh -k 5.18.0-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 5.18.0-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd.gz
root@slack:/usr/src/linux-5.18# mkinitrd -c -k 5.18.0-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/i
nitrd.gz
49039 bloków
/boot/initrd.gz created.
Be sure to run lilo again if you use it.
root@slack:/usr/src/linux-5.18# _

```

Figure 24 Generacja komendy

Ostatnim krokiem zostało nam edytować `/etc/lilo.conf` i uruchomić `lilo`

```

# Linux bootable partition config begins
image = /boot/vmlinuz
  root = /dev/sda1
  label = "Slackware 15.0"
  read-only
image = /boot/vmlinuz-nowametoda-5.18-smp
  root = /dev/sda1
  initrd = /boot/initrd.gz
  label = "nowa-metoda"
  read-only
# Linux bootable partition config ends

root@slack:/boot# lilo
Warning: LBA32 addressing assumed
Warning: Unable to determine video adapter in use in the present system.
Warning: Video adapter does not support VESA BIOS extensions needed for
display of 256 colors. Boot loader will fall back to TEXT only operation.
Added Slackware_15.0 *
Added nowa-metoda +
3 warnings were issued.
root@slack:/boot# _

```

Figure 25 Odpalenie lilo

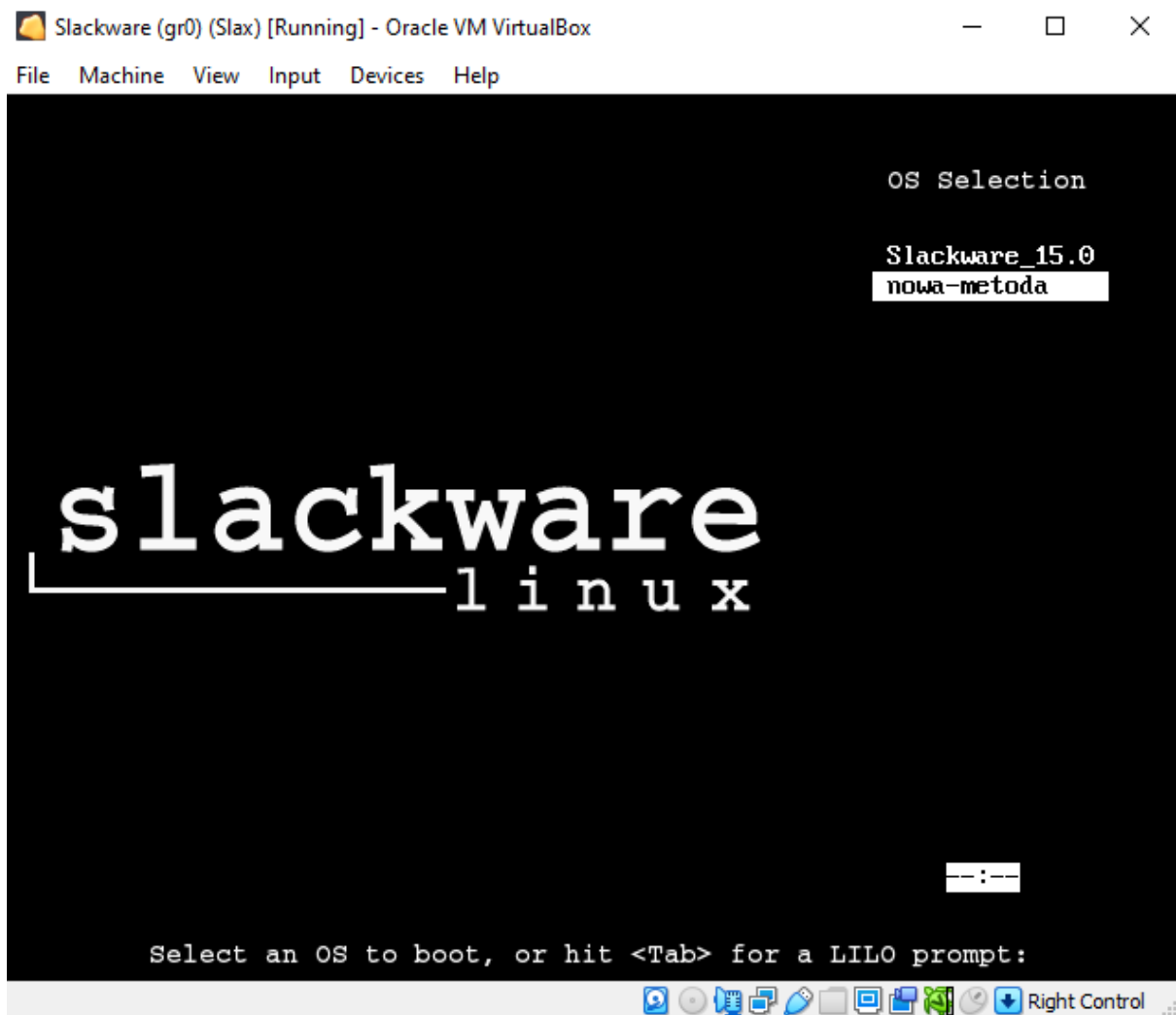



Figure 26 Nowa Metoda działająca





```
1.371125] zswap: loaded using pool lzo/zbud
1.374928] Key type ._fscrypt registered
1.374928] Key type .fscrypt registered
1.374928] Key type fscrypt-provisioning registered
1.377337] Btrfs loaded, crc32c=crc32c-intel, zoned=yes, fsverity=no
1.402998] Key type encrypted registered
1.466652] ata2.00: ATA-8: VBOX HARDDISK, 1.0, max UDMA/133
1.466652] ata2.00: 2097152 sectors, multi 128: LBA
1.484280] ata2.00: Read log 0x00 page 0x00 failed, Emask 0x1
1.485236] ata2.00: Security Log not supported
1.498956] ata2.00: Security Log not supported
1.504627] ata1.00: ATA-8: VBOX HARDDISK, 1.0, max UDMA/133
1.505627] ata1.00: 83886080 sectors, multi 128: LBA
1.521237] ata1.00: Read log 0x00 page 0x00 failed, Emask 0x1
1.521237] ata1.00: Security Log not supported
1.527495] ata1.01: ATAPI: VBOX CD-ROM, 1.0, max UDMA/133
1.527495] ata1.00: Security Log not supported
4.321164] floppy0: no floppy controllers found
4.329157] scsi 0:0:0:0: Direct-Access    ATA          VBOX HARDDISK    1.0  PQ: 0 ANSI: 5
4.341766] scsi 0:0:1:0: CD-ROM            VBOX          CD-ROM          1.0  PQ: 0 ANSI: 5
4.341766] sd 0:0:0:0: [sda] 83886080 512-byte logical blocks: (42.9 GB/40.0 GiB)
4.375205] sr 0:0:1:0: [sr0] scsi3-mmc drive: 32x/32x xa/form2 tray
4.375942] sd 0:0:0:0: [sda] Write Protect is off
4.375205] cdrom: Uniform CD-ROM driver Revision: 3.20
4.396117] sd 0:0:0:0: [sda] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
4.429327] scsi 1:0:0:0: Direct-Access    ATA          VBOX HARDDISK    1.0  PQ: 0 ANSI: 5
4.441155] sd 1:0:0:0: [sdb] 2097152 512-byte logical blocks: (1.07 GB/1.00 GiB)
4.447651] sd 1:0:0:0: [sdb] Write Protect is off
4.458086] sd 1:0:0:0: [sdb] Write cache: enabled, read cache: enabled, doesn't support DPO or FUA
4.468290] sda: sda1 sda2 sda3
4.489348] sd 0:0:0:0: [sda] Attached SCSI disk
4.492191] sdb: sdb1
4.513236] sd 1:0:0:0: [sdb] Attached SCSI disk
4.516410] Unstable clock detected, switching default tracing clock to "global"
4.516410] If you want to keep using the local clock, then add:
4.516410] "trace_clock=local"
4.516410] on the kernel command line
4.550193] Freeing unused kernel image (initmem) memory: 1036K
4.558417] Write protecting kernel text and read-only data: 21768k
4.558417] NX-protecting the kernel data: 10364k
4.559417] rodata_test: all tests were successful
4.559417] Run /init as init process
```




Figure 27 Odpalenie maszyny

```
slack login: root
Password:
Last login: Sat Jul  2 10:12:40 on tty1
Linux 5.18.0-smp.
root@slack:~# neofetch
```

```
root@slack:~#
```

```
OS: Slackware 15.0 i586 i686
Host: Oracle Corporation VirtualBox
Kernel: 5.18.0-smp
Uptime: 3 mins
Packages: 1497 (pkgtool)
Shell: bash 5.1.16
Resolution: 1024x768
Terminal: /dev/tty1
CPU: Intel i5-7300HQ (2) @ 0MHz
GPU: VMware SVGA II Adapter
Memory: 282MiB / 2000MiB
```



Figure 28 Potwierdzenie działającej nowej metody

## Podsumowanie

Obie metody są bardzo podobne do siebie. Co do czasu nie mogę skomentować, gdyż u mnie na tym laptopie i tych ustawieniach trwało to po prostu długo dla obu metod, na tyle, że zostawiłem laptopa i po prostu wróciłem do niego po jakimś czasie.

Stara metoda mi nie wyszła, ale po skończeniu nowej metody wychwyciłem swój błąd jaki popełniłem w starej metodzie (źle edytowałem lilo.conf) i wystarczyło by to jedynie poprawić aby zadziałała. Niestety usunąłem pliki po nieudanej kompilacji aby być w stanie zacząć od zera z nową metodą.