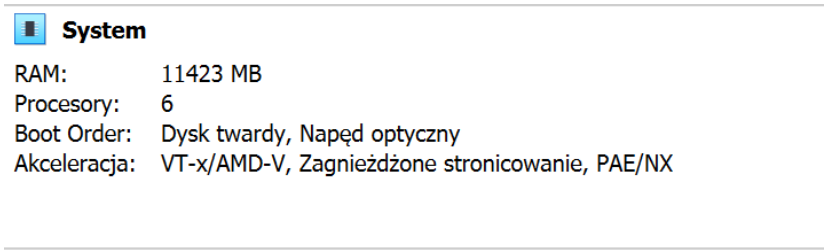
Sprawozdanie – Kompilacja Jądra

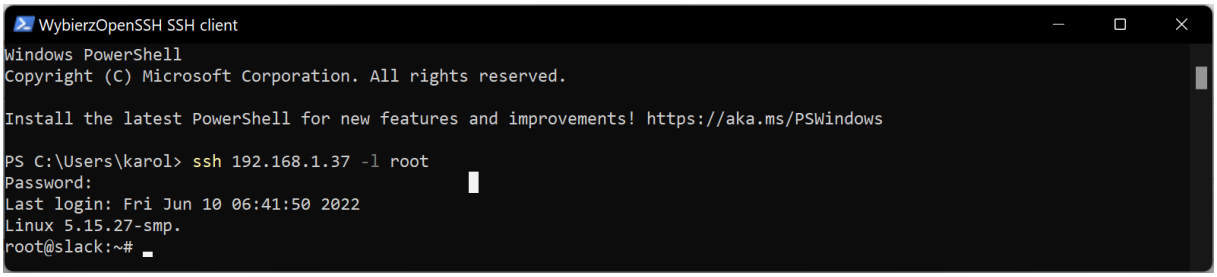
Karol Kula

Zanim przeszedłem do systemu linux ze względu na to, że mam dosyć mocny komputer najpierw zwiększyłem ilość ramu oraz rdzeni dla wirtualnej maszyny, dzięki temu oraz dzięki temu, że posiadam dysk ssd na złączu m2 kompilacja powinna przebiec bardzo szybko.

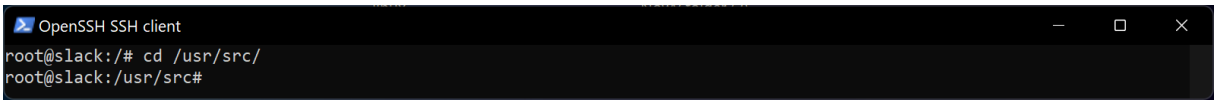


Pobranie kernela

Na początku oczywiście trzeba pobrać odpowiednią wersję jądra w tym celu odwiedziłem stronę www.kernel.org. W momencie pisania tego sprawozdania najnowsza stabilna wersją jądra to wersja **5.18.3**. Następnie uruchomiłem maszynę wirtualną jednak ze względu na to że do maszyny wirtualnej nie mogłem wkleić linka do jądra użyłem narzędzia PowerShell i za pomocą SSH zalogowałem się na konto root oczywiście wcześniej sprawdzając adres ip maszyny poleceniem *ip addr show.*

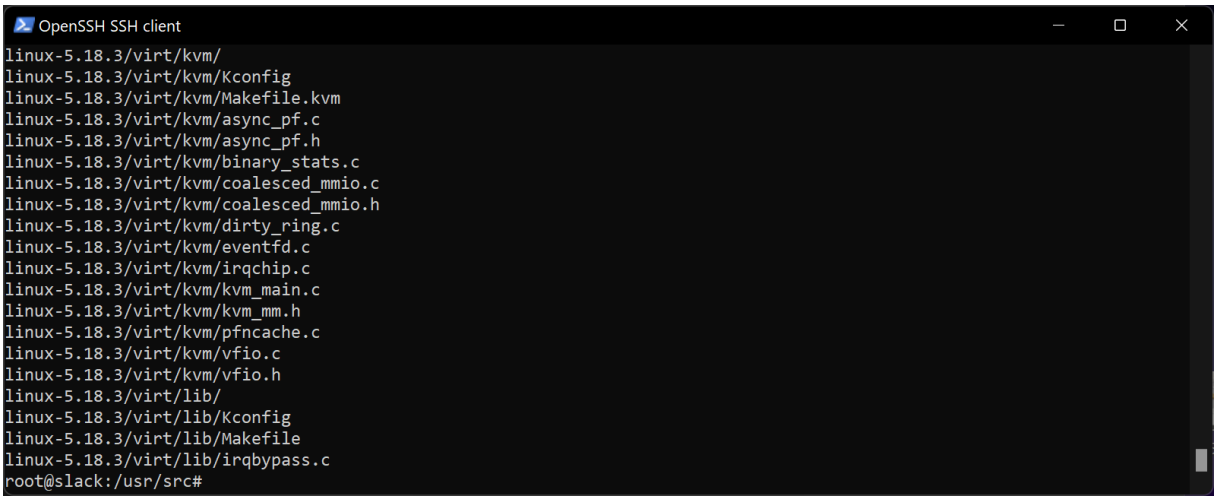


Następnie przeszedłem do katalogu */usr/src* dzięki poleceniu *cd*



Oraz użyłem polecenie wget z wcześniej skopiowanym linkiem, dzięki czemu rozpoczęło się pobieranie jądra

Następnie w celu rozpakowania pobranej paczki użyłem polecenia *tar -xvf linux-5.18.3.tar.xz* Końcówkę procesu rozpakowania widać na poniższym zrzucie ekranu.

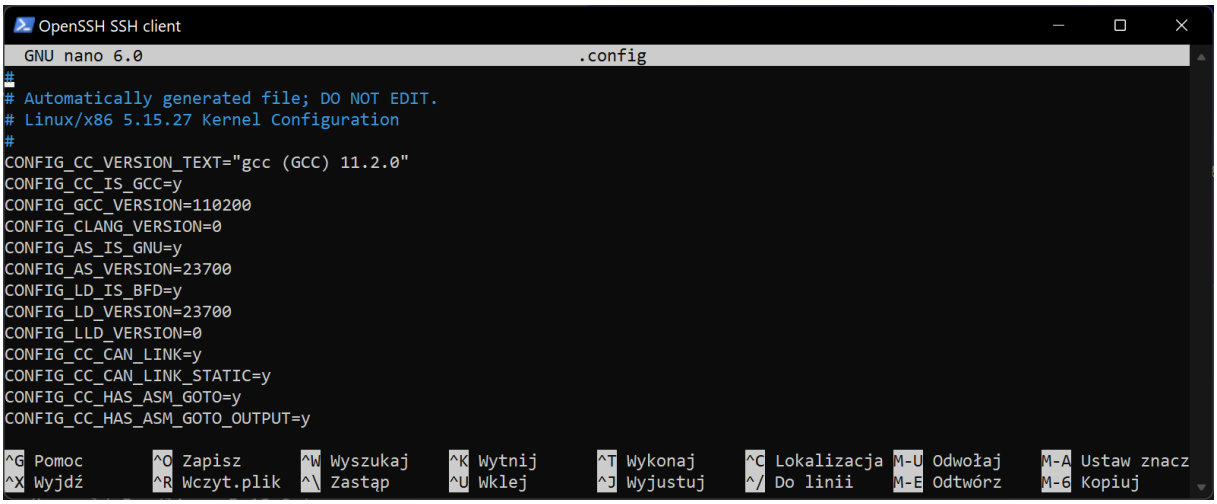


Metoda stara

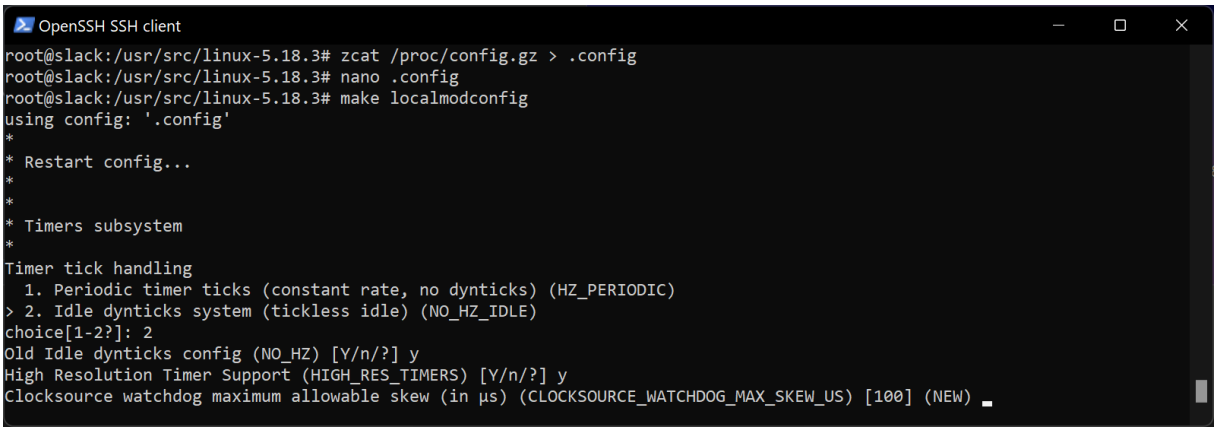
Po rozpakowaniu paczki pojawił się katalog *linux-5.18.3,* do którego wszedłem.

Następnie przekopiowałem aktualną konfigurację jądra do pliku .config

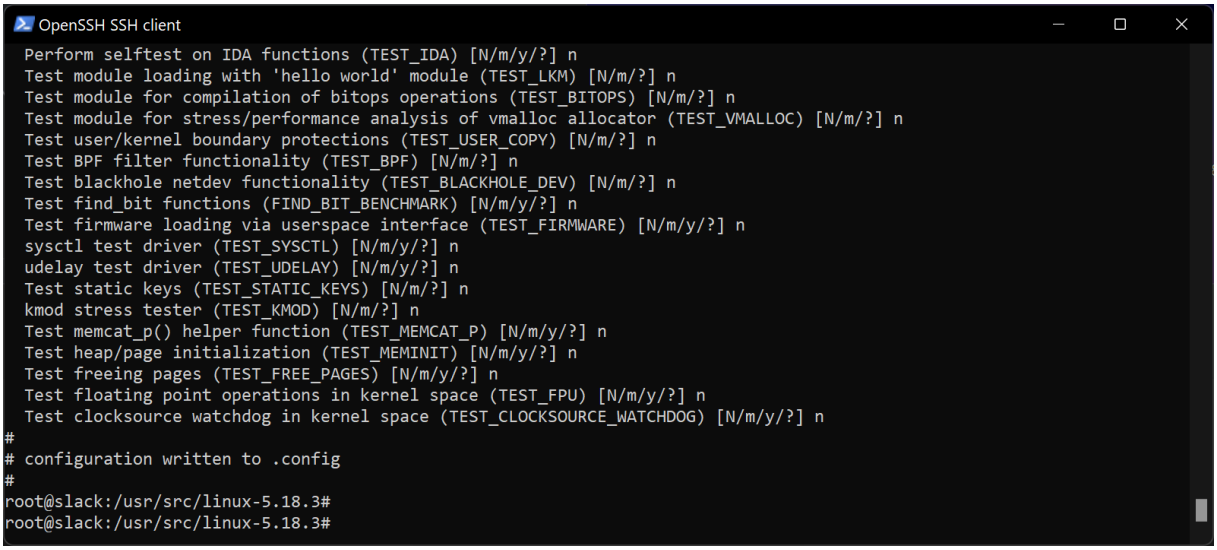
Oraz dla pewności sprawdziłem poleceniem *nano* czy plik na pewno został stworzony.



Następnie użyłem polecenia *make localmodconfig* w celu wygenerowania pliku.

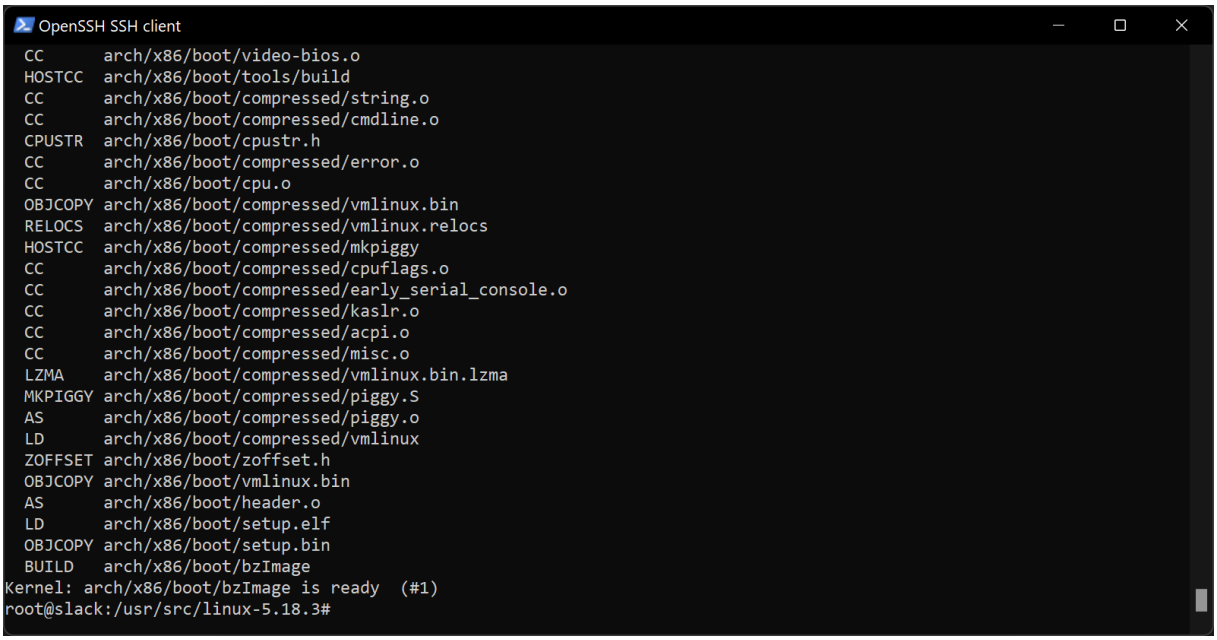


Następnie zostały wyświetlone pytania o poszczególne moduły jądra, zostawiłem wszystko domyślnie tak jak było na zajęciach.



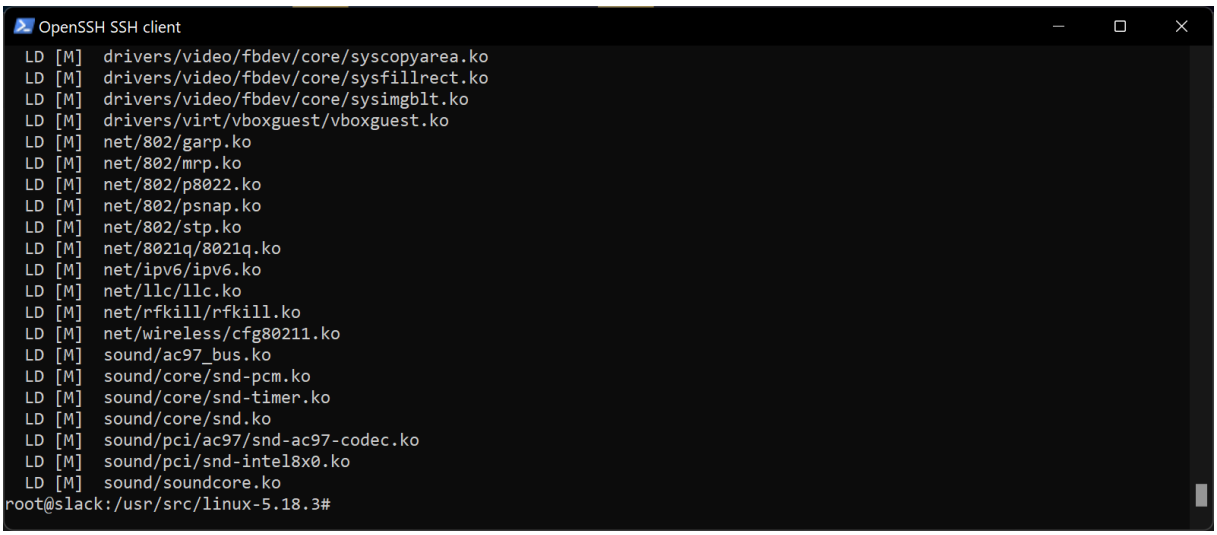
Następnie przeszedłem już do samej kompilacji jądra dzięki użyciu polecenia *make -j6 bzImage*, użyłem parametru *-j6* ze względu na to, że moja wirtualna maszyna wykorzystuje sześć rdzeni.

Tak jak przypuszczałem, dzięki udostępnieniu wirtualnej maszynie większej ilości zasobów, jądro skompilowało się całkiem szybko, a dokładnie to w 8 minut i 13 sekund.



Następnie przeszedłem do kompilacji modułów dzięki użyciu polecenia *make -j6 modules*

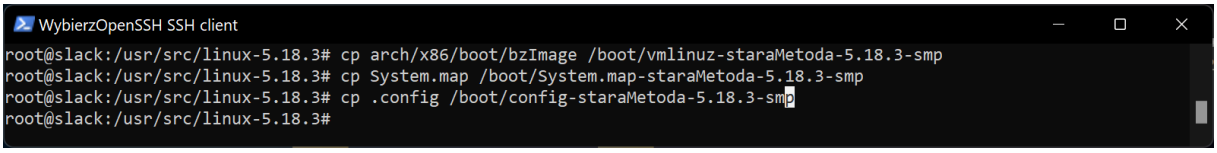
Skompilowanie modułów było bardzo szybkie dokładnie zajęło to 53 sekundy.



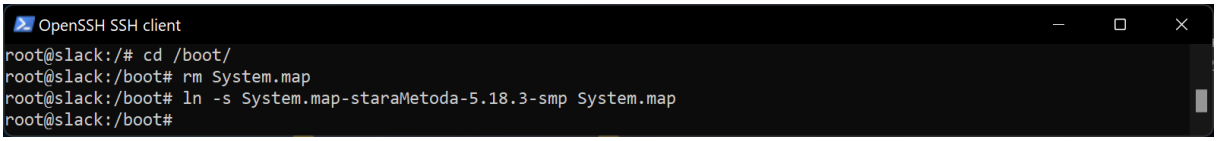
Następnie przeszedłem do instalacji modułów, dzięki użyciu polecenia *make modules_install*

Instalacja modułów była prawie natychmiastowa.

Następnie użyłem komendy *cp* do skopiowania plików potrzebnych do uruchomienia.



Następnie przeszedłem do tworzenia linku symbolicznego w systemie dla tablicy symboli kernela. W tym celu przeszedłem do katalogu */boot/* dzięku użyciu komendy *cd*, następnie usunąłem starą tablice symboli za pomocą komendy *rm*, i za pomocą komendy *ln* utworzyłem link symboliczny.



Po skopiowaniu plików i stworzeniu linku symbolicznego przeszedłem do generowania komendy, która stworzy ramdisk:

Używam tak wygenerowanej komendy tylko ze zmienią nazwą ostatniego parametru:

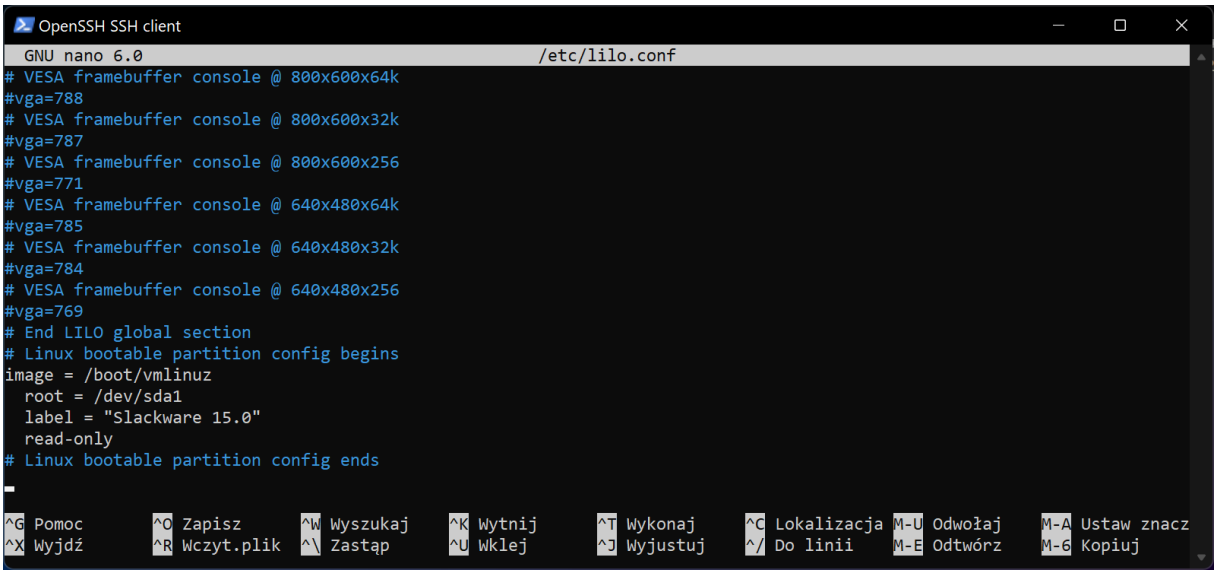
mkinitrd -c -k 5.18.3-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-staraMetoda-5.18.3-smp.gz

Jak widać na poniższym zrzucie wszystko przebiegło bez problemu.

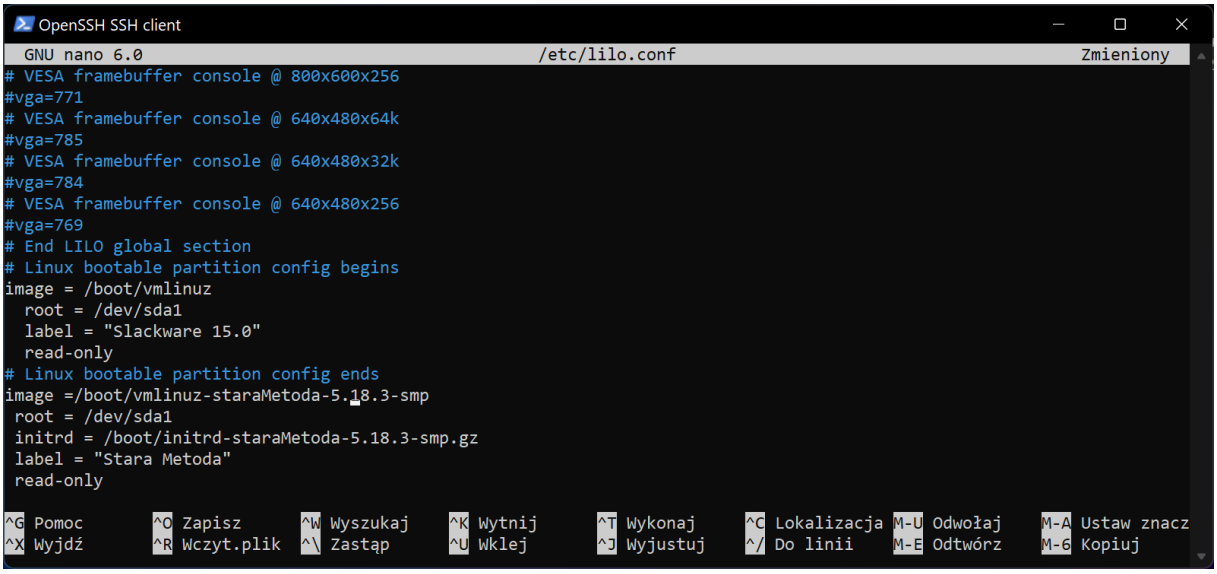
Aby można było skorzystać z tak przygotowanego kernela dodałem wpis LILO

Najpierw otworzyłem plik konfiguracyjny LILO za pomocą polecenia *nano*

Po przewinięciu pliku w dół, można zobaczyć miejsce gdzie dodaje się wpisy.

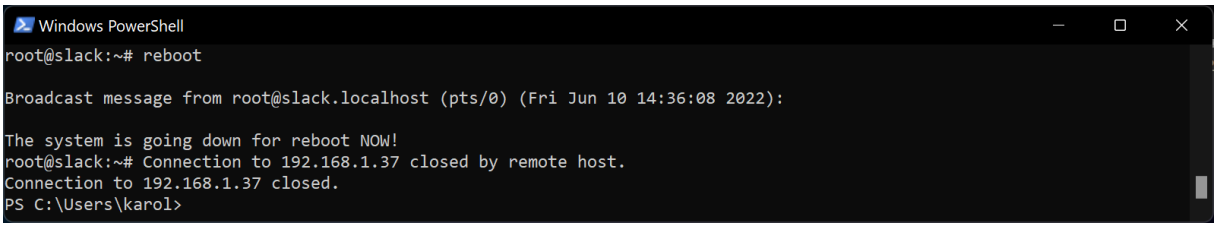


Następnie dodałem odpowiedni wpis dla metody starej.



Następnie zaktualizowałem LILO poleceniem *LILO*.

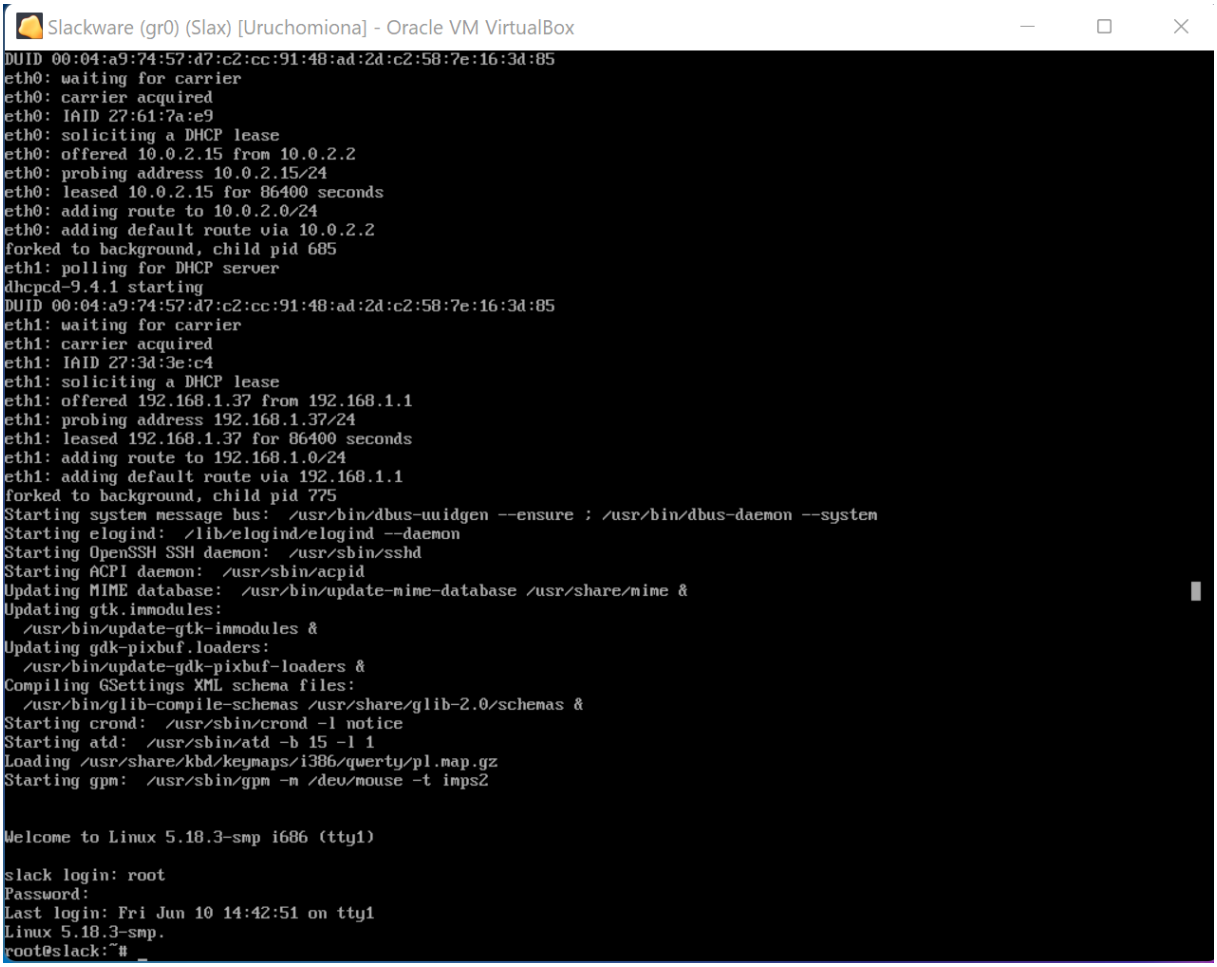
Oraz komendą *reboot* zrestartowałem maszynę.



Ponieważ zrestartowałem maszynę utraciłem z nią połączenie w powershellu ale przeszedłem bezpośrednio do okna wirtualnej maszyny.

Jak widać wpis został poprawnie dodany do LILO, i nowa wersja kernela jest możliwa do wyboru.

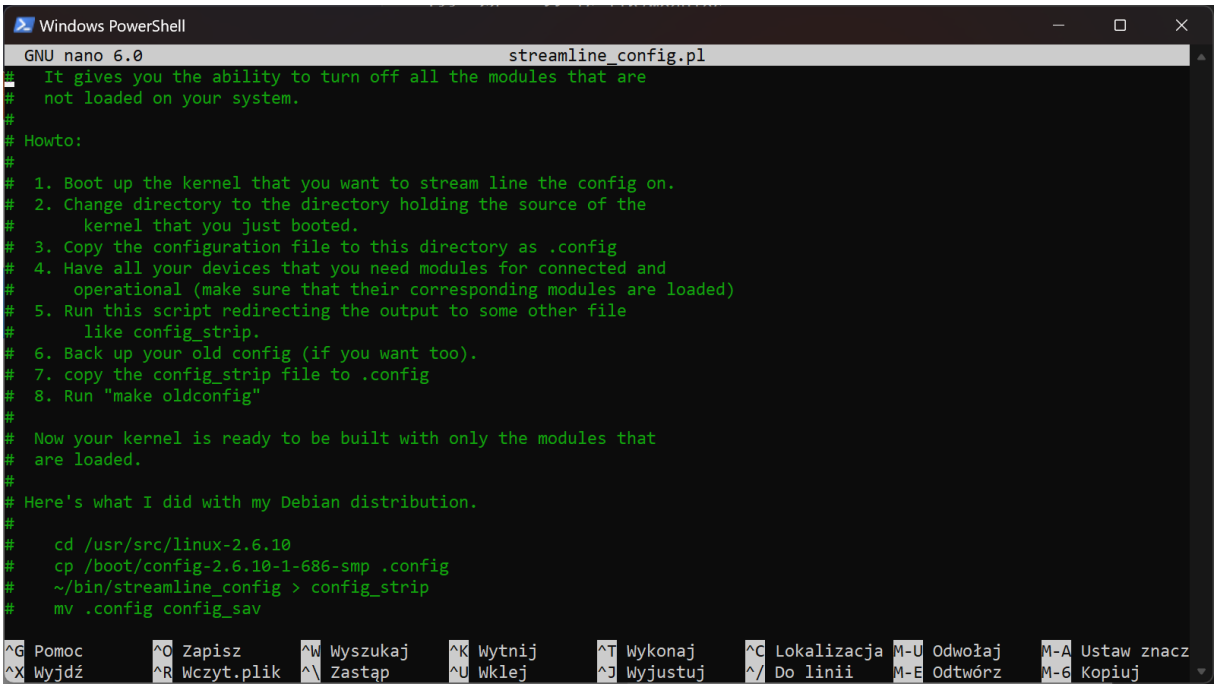
Następnie wybrałem wpis Stara_Metoda i na szczęścia cały system uruchomił się bez problemu.



Metoda Nowa

Ponieważ metoda nowa jest bardzo podobna do metody starej pozwolę sobie na nieco mniej szczegółowe opisy.

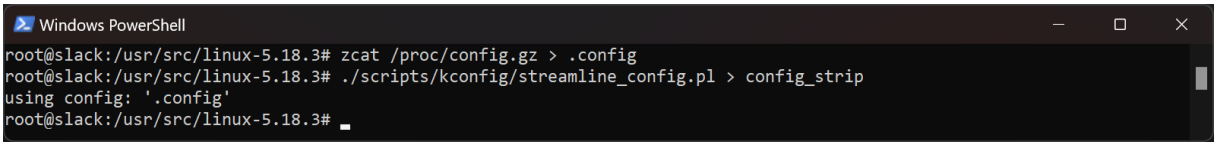
W folderze linux-5.18.3/scripts/kconfig znajduje się skrypt streamline_config.pl



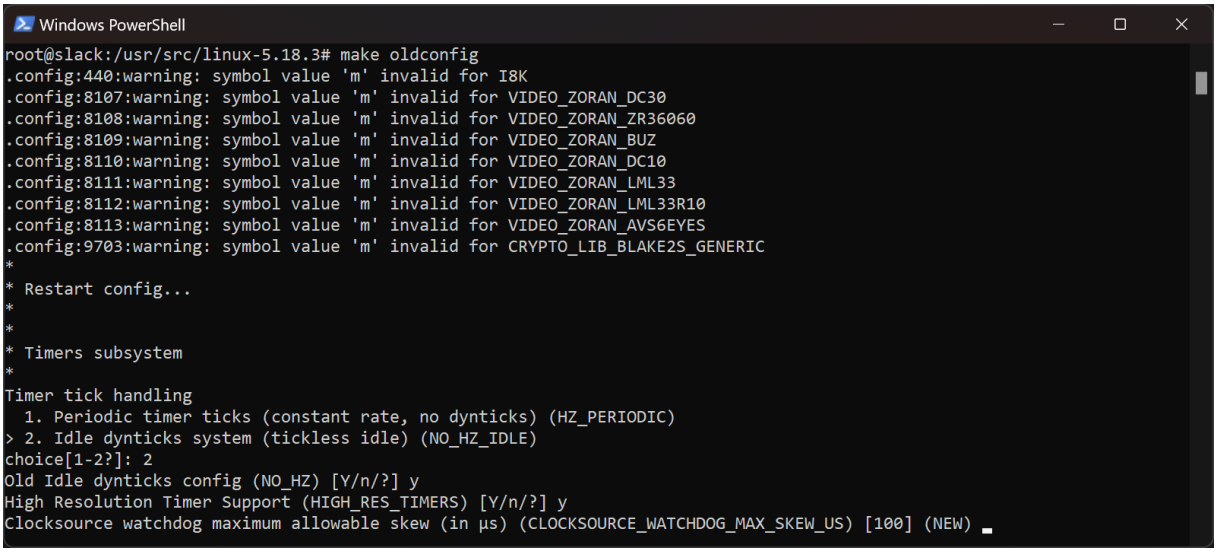
To właśnie tego skryptu używa się do kompilacji jądra nową metodą, w skrypcie jest opisane co należy robić krok po kroku.

Najpierw podobnie jak w metodzie starej stworzyłem plik konfiguracyjny:

A potem przeszedłem już do użycia skryptu.

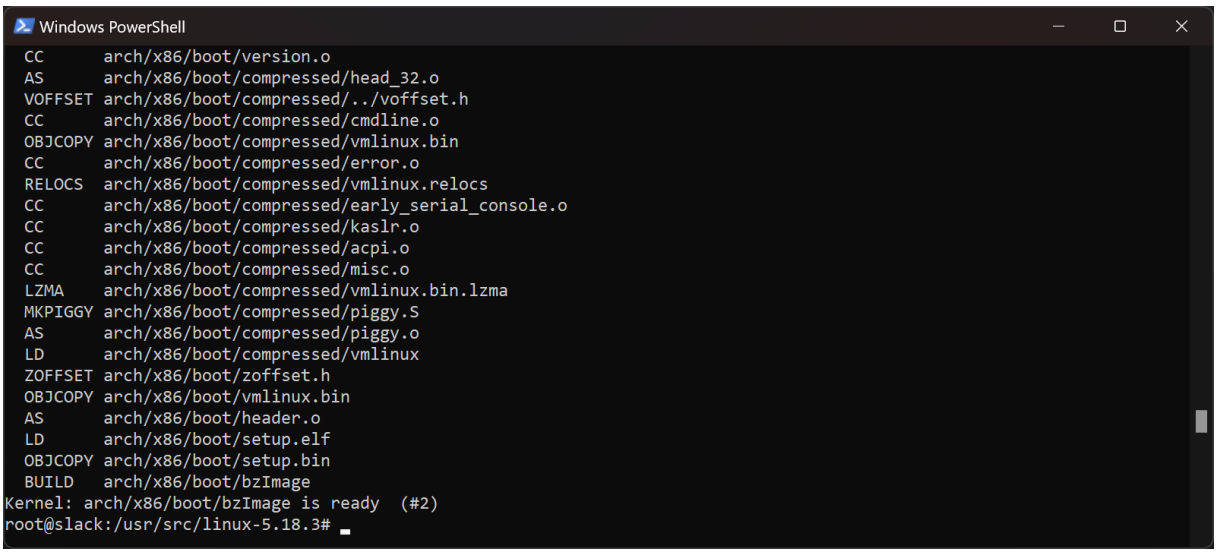


Po konfiguracji przeszedłem do wykonania komendy *make oldconfig.* I podobnie jak wcześniej wszystkie opcję zatwierdziłem przyciskiem enter.



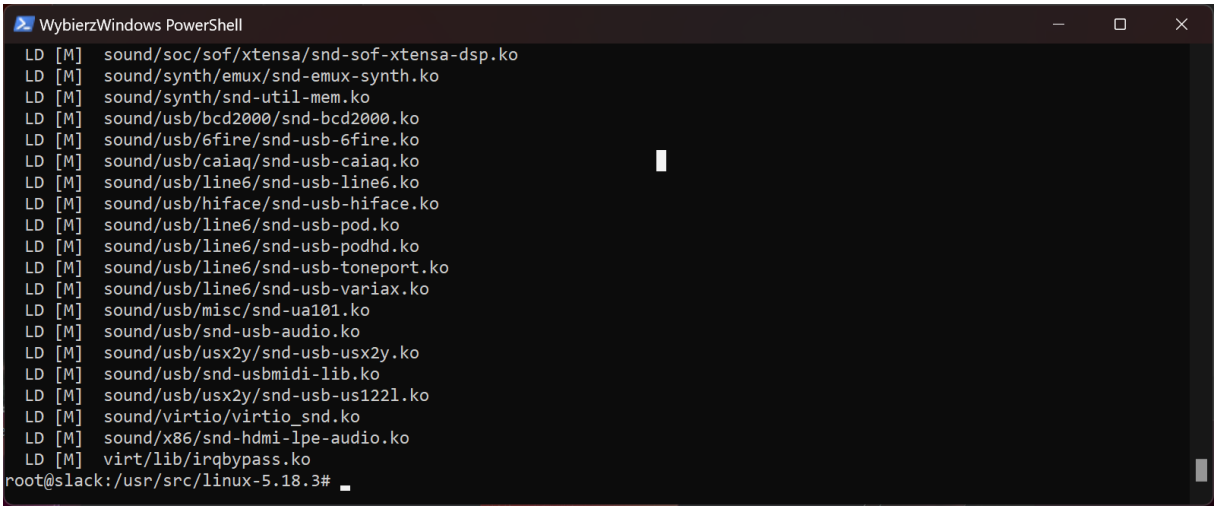
Po wykonaniu tej komendy przeszedłem do kompilacji jądra.

W przypadku nowej metody jądro skompilowało się w 11 minut i 15 sekund.



Tak jak poprzednio po kompilacji jądra trzeba skompilować moduły.

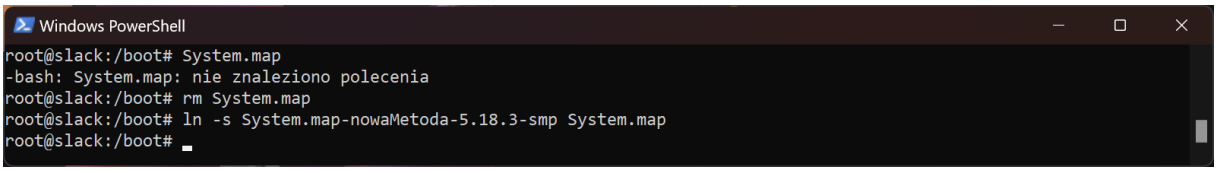
Ku mojemu zdziwieniu moduły kompilowały się 38 minut i 13 sekund



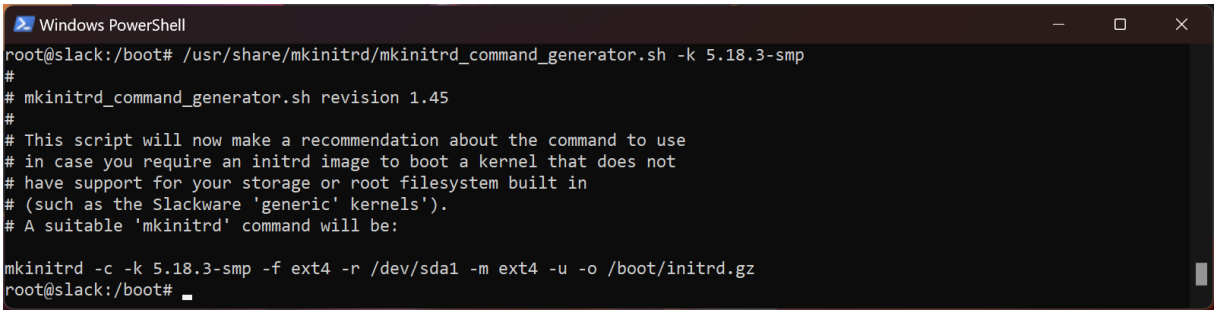
Następnie zainstalowałem moduły komendą *make modules_install* (instalowały się znacznie dłużej niż w metodzie starej)

Następnie użyłem komendy *cp* do skopiowania plików potrzebnych do uruchomienia.

Następnie przeszedłem do tworzenia linku symbolicznego w systemie dla tablicy symboli kernela. W tym celu przeszedłem do katalogu */boot/* dzięku użyciu komendy *cd*, następnie usunąłem starą tablice symboli za pomocą komendy *rm*, i za pomocą komendy *ln* utworzyłem link symboliczny.

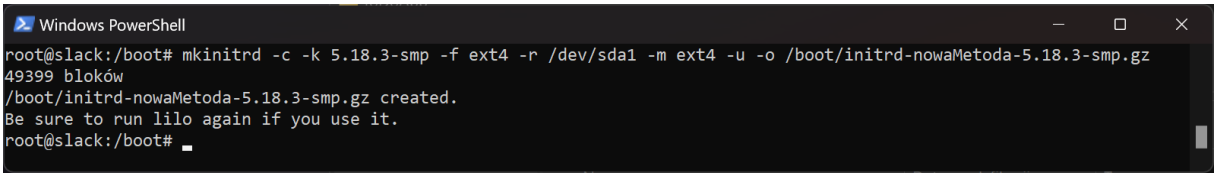


Po skopiowaniu plików i stworzeniu linku symbolicznego przeszedłem do generowania komendy, która stworzy ramdisk:

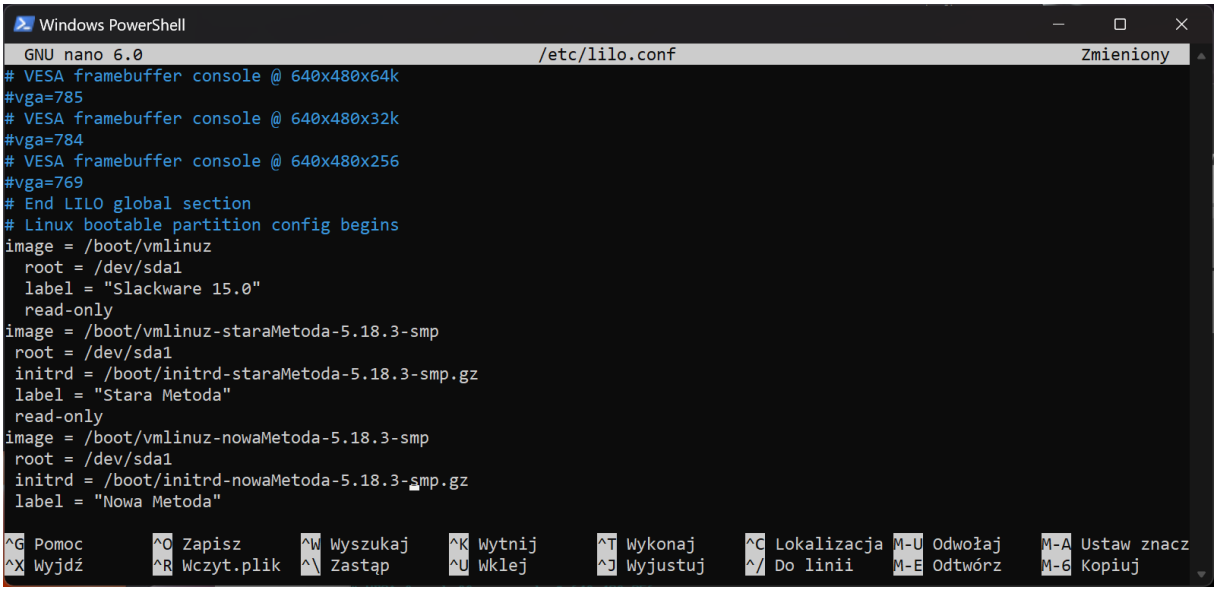


I użyłem tej komendy tylko modyfikując nazwę

mkinitrd -c -k 5.18.3-smp -f ext4 -r /dev/sda1 -m ext4 -u -o /boot/initrd-nowaMetoda-5.18.3-smp.gz



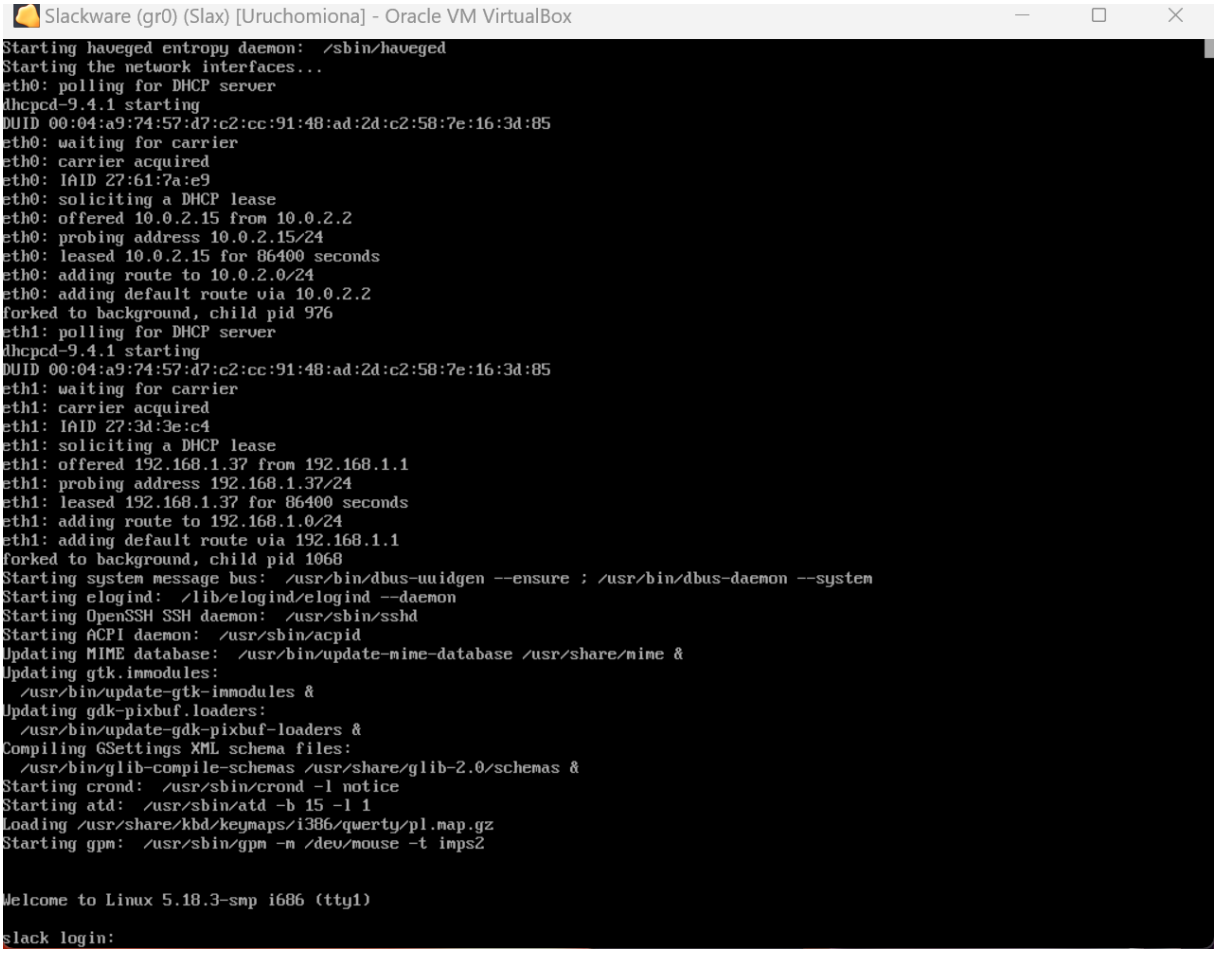
Ostatnim krokiem było oczywiście dodanie wpisu do lilo z tak przygotowanym kernelem



Oraz zaktualizowanie lilo poleceniem *lilo*

Po zresetowaniu maszyny wirtualnej widzimy że wszystko się w porządku i mamy do wyboru nową oraz starą metodę

A nowa metoda bez problemu się uruchamia:



Podsumowanie

Podczas kompilacji jądra dwiema metodami nie trafiłem na żadne problemy. Obie metody są tak naprawdę bardzo podobne jednak metoda nowa wydaje się lekko łatwiejsza ponieważ w skrypcie jest opisane co i jak należy robić. Mimo tego, że metoda nowa wydaję się prostsza to kompilacją modułów i samego jądra trwała tą metoda prawie 50 minut dłużej co może być nie pożądane gdy komuś się śpieszy. Co ciekawe zauważyłem, że w zasadzie za każdym razem system z kernelem skompilowanym nową metodą odpala się szybciej niż ten z kernelem skompilowanym metodą starą. Więc osoba stawiająca na wydajność systemu powinna wybrać metodę nową.