

Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI  
I TECHNIK INFORMACYJNYCH



Instytut Automatyki i Informatyki Stosowanej

# Praca dyplomowa inżynierska

na kierunku Informatyka  
w specjalności Systemy Informacyjno-Decyzyjne

Laboratorium systemów operacyjnych w środowisku systemu Linux

**Jakub Wesołowski**

Numer albumu 276921

promotor  
dr inż. Tomasz Jordan Kruk

Warszawa 2020



## **Streszczenie**

**Tytuł:** Laboratorium systemów operacyjnych w środowisku systemu Linux

Niniejsza praca dotyczy stworzenia środowiska pracy dla studenta nad zadaniami laboratorium przedmiotu Systemy Operacyjne, które ma zapewnić wygodniejsze oraz bardziej wydajne wykonywanie zadań laboratoryjnych. To środowisko powstaje w celu bycia wdrożonym jako element zajęć ulepszając starą formę prowadzenia laboratorium. Największą zmianą ma być dostęp sieciowy do maszyny wirtualnej z systemem Minix pozwalający na kopiowanie oraz przenoszenie plików na drugą maszynę z systemem Ubuntu. Dzięki takiemu rozwiązaniu student będzie mógł wykorzystać nowoczesne narzędzia do edycji plików oraz inne oprogramowanie dostępne na systemie Ubuntu do pracy na laboratorium, nie jest on zmuszony do korzystania wyłącznie z oprogramowania dostarczonego wraz z systemem Minix jak dotychczas.

**Słowa kluczowe:** Minix, Ubuntu, maszyna wirtualna, system operacyjny, karta sieciowa, OpenSSH



## Spis Treści

<b>1. Wstęp</b> .....	<b>1</b>
1.1. Cel pracy.....	1
1.2. Układ pracy.....	2
<b>2. Wprowadzenie do Miniksa</b> .....	<b>3</b>
<b>3. Wersje Miniksa</b> .....	<b>5</b>
3.1. Minix 2.0.3.....	5
3.2. Minix 2.0.4.....	5
3.3. Minix 3.1.....	5
3.4. Wybór wersji Miniksa .....	6
<b>4. Instalacja Miniksa 2.0.3</b> .....	<b>9</b>
4.1. Przygotowanie dyskietki instalacyjnej .....	9
4.2. Przygotowanie maszyny wirtualnej w środowisku VMWare .....	9
4.3. Instalacja systemu operacyjnego Minix 2.0.3 .....	14
4.4. Instalacja Miniksa 2.0.3 w środowiskach innych niż VMWare .....	21
4.5. Rozwiązywanie problemów z dyskietką, „fs: I/O error on device” .....	22
<b>5. Sterownik karty sieciowej AMD Lance</b> .....	<b>23</b>
5.1. Test sterownika dla Miniksa 2.0.2.....	23
5.2. Test sterownika dla Miniksa 2.0.4.....	24
5.3. Konwersja sterownika karty sieciowej AMD Lance na Miniksa 2.0.3 .....	25
<b>6. Oprogramowanie SSH na Miniksa 2.0.3</b> .....	<b>27</b>
6.1. OpenSSH 1.2.3.....	27
6.2. OpenSSH 2.1.1.....	29
<b>7. Środowisko do pracy na laboratorium</b> .....	<b>33</b>
7.1. Zawartość płyty CD.....	34
<b>8. Instalacja środowiska do pracy na laboratorium</b> .....	<b>35</b>
8.1. Instalacja pod VMWare .....	35
8.2. Instalacja pod VirtualBoxem .....	35
8.3. Instalacja pod Qemu .....	36
8.4. Konfiguracja systemu Ubuntu.....	36
<b>9. Podsumowanie</b> .....	<b>39</b>

<b>10. Bibliografia.....</b>	<b>41</b>
<b>11. Spis rysunków .....</b>	<b>43</b>
<b>12. Spis tabel .....</b>	<b>45</b>

## 1. Wstęp

Na przedmiocie Systemy Operacyjne studenci poznają podstawowe własności systemów operacyjnych. W ramach laboratorium tego przedmiotu student realizuje zadania w środowisku systemów operacyjnych Minix i Linux. Student uczy się jak skompilować jądro systemu, z czego ono się składa, pozyskuje wiedzę na temat wywołań systemowych, szeregowania procesów, semaforów, zarządzania pamięcią, działania systemu plików. Wiedza ta jest podstawą do wykonania zadań, które wymagają od studenta zaimplementowania nowych algorytmów szeregowania procesów, zarządzania pamięcią, czy skonstruowania prostego systemu plików. Mniejsza część zadań laboratoryjnych, głównie związana z semaforami oraz monitorami odbywa się na systemie Ubuntu opartym o jądro Linux. Reszta z zadań odbywa się na systemie Minix.

Są to w większości zadania wymagające od studenta edycji kodu źródłowego jądra systemu. Na zajęciach studenci korzystają w tym celu z systemu Minix w wersji 2.0.3. System ten został stworzony w celach edukacyjnych, więc jest dobrym obiektem do nauki działania jądra systemu operacyjnego. Jednocześnie jest on także mniej skupiony na funkcjonalności i wygodzie, oraz już całkiem stary, wersja 2.0.3 pochodzi z roku 2001. Zawiera on wyłącznie najprostsze edytory tekstu, z których żaden nie obsługuje podświetlania składni. System wspiera jedynie rozdzielczość ekranu 640x480. Minix posiada wsparcie dla komunikacji sieciowej, ale jest ona dodatkowym elementem systemu, który trzeba zainstalować oddzielnie. Dystrybucja Miniksa dotychczas używana na laboratorium nie była skonfigurowana do obsługi sieci. Z tego powodu nie istniała łatwa metoda na przesyłanie plików i komend między systemem Minix a innymi systemami operacyjnymi. Wymienione powyżej mankamenty powodują, że praca nad zadaniami laboratoryjnymi z Systemów Operacyjnych jest mało wygodna. Szukanie błędów w programie na małym ekranie, z mało czytelną czcionką oraz bez podświetlania składni może być uciążliwe dla studenta. W celu wysłania gotowego rozwiązania prowadzącemu laboratorium student często musiał przepisywać wprowadzone przez siebie zmiany jądra systemu Minix do raportu.

Wyżej wymienione trudności to tylko część z problemów, z którymi musi zmierzyć się student chcący zaliczyć laboratorium z przedmiotu Systemy Operacyjne. Zauważono potrzebę zmodernizowania środowiska pracy studentów tak, aby student mógł pracować w warunkach wygodniejszych, w przygotowanym środowisku, które będzie użyteczne, wygodne oraz przenośne. Rozwiązanie to jednocześnie nie zamyka tej tematyki, wynik pracy inżynierskiej może być dalej rozwijany przez innych studentów lub pracowników wydziału.

### 1.1. Cel pracy

Celem pracy jest stworzenie nowoczesnego środowiska do pracy nad zadaniami laboratorium przedmiotu Systemy Operacyjne, składającego się z dwóch maszyn wirtualnych: z systemem Minix oraz z systemem Ubuntu. Maszyna z Miniksem ma zawierać wsparcie dla obsługi sieci w celu komunikacji z maszyną Ubuntu, pozwalać na przesyłanie plików między maszynami. Maszyna z Ubuntu ma zapewnić uczestnikom laboratorium dostęp do nowoczesnych programów, np. do edycji plików kodu źródłowego jądra, będzie ona głównym środowiskiem pracy. Całość ma zostać przekazana studentom wraz z instrukcjami instalacji. Rozwiązanie powinno być udokumentowane, łatwe w instalacji i konfiguracji, przenośne oraz powinno wspierać wiele oprogramowań służących do wirtualizacji.

## **1.2. Układ pracy**

Rozdziały od 3. do 6. wprowadzają do tematyki pracy oraz opisują pracę wykonaną nad stworzeniem takiego środowiska, rozdział 7. opisuje wynikowe środowisko pracy, a rozdział 8. opisuje proces jego instalacji. Rozdział 9. zawiera podsumowanie wraz z perspektywami rozwoju pracy. Na końcu pracy znajduje się bibliografia oraz spis rysunków i tabel.



## 2. Wprowadzenie do Miniksa

Początek mojej pracy zacznę od krótkiego wprowadzenia do systemu Minix w celu lepszego zrozumienia podjętych w trakcie pracy decyzji oraz postawionych celów.

System Minix jest systemem uniksopodobnym stworzonym przez prof. Andrewa S. Tanenbauma. Minix powstał głównie w celach edukacyjnych. Pierwsza wersja tego systemu z numerem 1 została wydana w 1987 r. W tym samym roku wydana została książka „Operating Systems: Design and Implementation” napisana przez prof. Tanenbauma oraz dr Alberta S. Woodhulla. Jest ona książką naukową opisującą działanie systemu operacyjnego na przykładzie Miniksa. W książce wydrukowany został także pełny kod źródłowy systemu w wersji 1.0 [7].

W roku 1996 wydany został system Minix w wersji 2.0. Ponownie wraz z systemem pojawiło się drugie wydanie książki „Operating Systems: Design and Implementation”, tym razem opierające się na nowej wersji Miniksa. Wraz z książką załączona została płyta CD z kodem źródłowym jądra.

System Minix 2 był dalej rozwijany, wydane zostały trzy aktualizacje systemu: wersja 2.0.2 w 1998 roku, wersja 2.0.3 w 2002 roku oraz wersja 2.0.4 w 2006 roku [3]. Kolejne rozwinięcia wersji 2.0 wprowadziły poprawki błędów, niektóre moduły jądra zostały przepisane na nowo, zmieniono nazewnictwo urządzeń nośników danych, czy też poprawiono obsługę sieci [6].

Minix 3.1 został opublikowany w 2005 roku [8]. Trzecie wydanie książki „Operating Systems: Design and Implementation” wydane zostało w 2006 roku. Największą różnicą w porównaniu z poprzednimi wersjami jest odejście od edukacyjnego charakteru systemu, a położenie nacisku na system produkcyjny oraz skupienie się na systemach wbudowanych [7].

Należy pamiętać, że poszczególne wydania książki opierają się na kodzie Miniksa z pierwszych wydań poszczególnych wersji systemu. Kolejne aktualizacje systemu zmieniły kod źródłowy i zdezaktualizowały część książki.

W celu stworzenia środowiska pracy, które jest produktem tej pracy inżynierskiej musiałem zdecydować, której wersji systemu Minix użyć. Wybór ten opisuję w następnym rozdziale.



### 3. Wersje Miniksa

Na laboratorium przedmiotu Systemy Operacyjne studenci korzystali dotychczas z Miniksa w wersji 2.0.3. Wraz z opiekunem musieliśmy zastanowić się, czy chcemy kontynuować to laboratorium na tej wersji, czy też przenieść się na wersję nowszą. Przy wyborze najważniejszym kryterium była obsługa sieci, posiadane oprogramowanie oraz prostota i rozmiar zajmowany przez system.

#### 3.1. Minix 2.0.3

Jedną z możliwości jest pozostanie przy Miniksie w wersji 2.0.3. Wersja ta została wydana w 2002 roku. W porównaniu do wersji 2.0.2 wprowadza wsparcie dla DHCP zastępując przestarzałe RARP, zmienia nazewnictwo urządzeń pamięci masowej, rozszerza limit pamięci dyskowej z 4GB do 2TB, zastępuje starą powłokę poleceń powłoką ASH z BSD, zapewnia podstawowe wsparcie dla pamięci swap [4]. Podobnie jak wszystkie wersje Miniksa 2, Minix 2.0.3 posiada obsługę sieci, która jest standardowo wyłączona, i nie była do tej pory na laboratorium przedmiotu Systemy Operacyjne do niej skonfigurowana. Obsługa sieci oraz sterowniki kart sieciowych są kompilowane wraz z jądrem, należy skonfigurować jądro do ich obsługi, a następnie je zbudować [2]. Wśród programów na Miniksie 2.0.3 możemy znaleźć podstawowe narzędzia sieciowe tj. klient DHCP, klienty oraz serwery programów telnet, RSH i FTP. Przy ustawieniu maszyny wirtualnej z Miniksem 2.0.3 z działającą obsługą sieci największym problemem okazują się sterowniki kart sieciowych wbudowane w system. System Minix 2.0.3 obsługuje następujące karty sieciowe: DP8390, Western Digital WD80x3, Novell NE1000/2000 oraz 3Com Etherlink II 3C503. Większość z popularnego oprogramowania do wirtualizacji (VirtualBox, VMWare oraz Hyper-V) nie obsługuje emulacji wyżej wymienionych kart sieciowych.

#### 3.2. Minix 2.0.4

Po Miniksie 2.0.3 została wydana jeszcze jedna aktualizacja z numerem 2.0.4 w 2006 roku. Aktualizacja wprowadza niewiele zmian, m.in. możliwość wyłączenia komputera z poziomu monitora inicjacji systemu (ang. boot monitor), dodaje obsługę karty sieciowej Realtek RTL8139 oraz wprowadza liczne refaktoryzacje w kodzie jądra [6]. Z punktu widzenia mojej pracy jedyną ważną zmianą w tej wersji jest dodanie sterownika nowej karty sieciowej.

#### 3.3. Minix 3.1

Kolejną z możliwości jest wybranie systemu Minix w wersji 3.1. Minix 3.1 jest nowocześniejszym i prostszym w użyciu. System instalowany jest z płyty live CD, standardowo posiada wbudowaną obsługę sieci, obsługuje wiele rodzin kart sieciowych. Instalacja systemu przebiega przez program instalacyjny, w którym użytkownik może wybrać m.in. swój interfejs sieciowy. Po instalacji system jest w pełni gotowy do użycia i komunikacji w sieci. System zawiera wywołanie systemowe select, którego brakuje na Miniksie 2. Posiada on także szereg nowocześniejszych programów, m.in. GCC, G++, Emacs, Vim, Python, Perl, narzędzia GNU [1]. Następne wersje Miniksa 3 wprowadzają menedżer pakietów, który dalej rozszerza wybór oprogramowania [8].

Minix 3 z pewnością jest systemem wygodniejszym w użyciu w porównaniu do wersji Miniksa 2. Twórcy systemu włożyli dużo pracy, aby uczynić ten system użytecznym dla codziennego użytkownika, ale wraz z rozwojem system stał się bardziej skomplikowany. Jego jądro, o ile nadal bardzo proste w porównaniu do jądra Linuxa czy jąder systemów rodziny BSD, znacznie zwiększyło swój rozmiar poprzez coraz większe dążenie do kompatybilności z tymi jądrami. Ma to także wpływ na czas

kompilacji samego systemu oraz jego wagę na dysku. Wersja 2.0.3 zajmuje na tyle mało pamięci na dysku, że studenci mogli sobie pozwolić na tworzenie kopii zapasowych systemu poprzez kopiowanie obrazu dysku maszyny wirtualnej. W przypadku wersji 3 baza tych kopii szybko nabrałaby dużej powierzchni pamięci komputera. Dłuższy czas kompilacji sprawia, że student będzie spędzał więcej czasu czekając na zbudowanie jądra. Czas ten może się nawarstwiać przy częstych zmianach i poprawkach. Kolejnym z problemów jest treść zadań laboratoryjnych, które zostały przygotowane konkretnie pod Miniksa w wersji 2.0.3. Jeśli miałbym zdecydować się na wybór Miniksa 3 do realizacji środowiska pracy na laboratorium, trzeba by było przepisać wszystkie zadania laboratoryjne na Miniksa 3. Takie rozwiązanie mogłoby okazać się bardzo niekorzystne zarówno dla studentów, jak i prowadzących laboratorium. Studenci nie mogliby polegać na dotychczas dostarczonej i sprawdzonej na przestrzeni lat wiedzy, a prowadzący laboratorium musieliby przestudiować nowy kod źródłowy oraz nowo sporządzone treści zadań w na tyle wysokim stopniu, aby móc pomóc studentom w razie problemów czy na konsultacjach.

### 3.4. Wybór wersji Miniksa

Pierwszym z kryteriów wyboru wersji Miniksa do realizacji środowiska pracy na laboratorium jest działająca obsługa sieci. Do jej działania potrzebna jest obsługa karty sieciowej emulowanej przez maszynę wirtualną w jądrze systemu. W celu porównania dostępnych sterowników interfejsów kart sieciowych na poszczególnych wersjach Miniksa z wspieranymi interfejsami sieciowymi w programach do wirtualizacji przygotowałem poniższe dwie tabele:

Karta sieciowa	Minix 2.0.0	Minix 2.0.2	Minix 2.0.3	Minix 2.0.4	Minix 3.1
DP8390	+	+	+	+	+
Western Digital WD80x3	-	+	+	+	+
Novell NE1000/2000	-	+	+	+	+
3Com Etherlink II 3C503	-	+	+	+	+
Realtek RTL8139	-	-	-	+	+
Realtek RTL8029	-	-	-	-	+
Intel Pro/100	-	-	-	-	+
AMD Lance	-	-	-	-	+

Tabela 1: Wsparcie dla kart sieciowych w poszczególnych wersjach systemu Minix

Karta sieciowa	VMWare	VirtualBox	Qemu	Bochs	Hyper-V
DP8390	-	-	-	-	-
Western Digital WD80x3	-	-	-	-	-
Novell NE1000/2000	-	-	-	+	-
3Com Etherlink II 3C503	-	-	-	-	-
Realtek RTL8139	-	-	+	-	-
Realtek RTL8029	-	-	-	-	-
Intel Pro/100	+	+	+	+	-
AMD Lance	+	+	+	-	-

Tabela 2: Wsparcie emulacji kart sieciowych przez oprogramowanie do wirtualizacji

Z powyższych tabel można wywnioskować, że obsługa sieci na Miniksie 3.1 powinna działać na VMWare, VirtualBox, Qemu i Bochs. Przetestowałem i potwierdziłem jej działanie na oprogramowaniu VMWare i VirtualBox. Po instalacji systemu sieć działa, nie jest potrzebna żadna dodatkowa konfiguracja.

Minix 2.0.3 według tabel kompatybilny jest jedynie z kartą sieciową emulowaną przez oprogramowanie Bochs. Wersja 2.0.4 Miniksa wprowadza wsparcie dla karty sieciowej Realtek RTL8139, która może być emulowana przez Qemu. Podczas testów nie udało mi się doprowadzić do działania sieci na maszynie wirtualnej Qemu z Miniksem 2.0.4. Oznacza to, że żadna z wersji Miniksa 2 nie oferuje prostej obsługi sieci, wybierając Miniksa 2 trzeba będzie rozszerzyć wsparcie Miniksa o dodatkową kartę sieciową wspieraną przez większość z oprogramowania do wirtualizacji.

Drugim z kryteriów jest prostota i wygoda z korzystania z systemu w ramach laboratorium. Minix 3.1 zawiera większy wybór oprogramowania w porównaniu do Miniksa 2, ale jest za to o wiele cięższy. Świeżo zainstalowana maszyna Miniksa 3.1 może zajmować ponad 300MB miejsca na dysku, w porównaniu do Miniksa 2, który zajmuje nieco ponad 22MB. Niska waga maszyny może okazać się przydatna przy tworzeniu kopii zapasowych systemu, szczególnie w przypadku wprowadzania zmian w jądrze, gdzie kompilacja przeedytowanego jądra może spowodować nieuruchamianie się systemu. Jądro systemu Minix 2.0.3 jest także o wiele lżejsze od jądra Miniksa 3.1, przez co kompiluje się o wiele szybciej, co wpływa na wygodę podczas pracy nad zadaniami laboratoryjnymi.

Samo jądro jest bardzo ważnym kryterium przy wyborze wersji do realizacji mojej pracy inżynierskiej. Dotychczas na laboratorium używano wersji Miniksa 2.0.3, a więc wszystkie zadania laboratoryjne do realizacji na Miniksie pisane były pod wersję 2.0.3. Zmiany w jądrze spowodowane przez nowsze wersje Miniksa mogłyby zdezaktualizować treść zadań laboratoryjnych. Stworzyłyby to wymóg przepisania tych zadań na nowszą wybraną przeze mnie wersję Miniksa. Mogłoby to się okazać kłopotliwe i wymusić także na prowadzących laboratorium dokładne przestudiowanie zadań oraz jądra nowszej wersji systemu operacyjnego.

Biorąc pod uwagę powyższe kryteria zdecydowałem się wybrać Miniksa 2.0.3 do realizacji mojej pracy inżynierskiej, czyli zachować wersję, z której dotychczas korzystaliśmy na laboratorium. Minix 3.1 posiada zaletę wysokiej kompatybilności wsparcia sieci na wielu systemach do wirtualizacji, ale poza tym zdaje się wprowadzać wiele problemów i wymusza zmienienie samej formy laboratorium Systemów Operacyjnych poprzez przepisanie zadań laboratoryjnych na ten system. Minix 2.0.4 mógłby być dobrym kandydatem, jeżeli sterownik karty sieciowej Realtek RTL8139 w jądrze Miniksa rozpoznawałby interfejs emulowany przez Qemu. Minix 2.0.4 nie oferuje jednak nic ważnego z punktu widzenia realizacji tej pracy inżynierskiej w porównaniu do Miniksa 2.0.3.

Minix 2.0.3 co prawda nie posiada sterowników wspieranych przez przetestowane oprogramowanie do wirtualizacji (z wyjątkiem Bochs), ale pozwoli na zachowanie dotychczasowej formy realizacji zadań laboratoryjnych. Podczas testów nie udało mi się skonfigurować działającej maszyny Bochs z Miniksem 2.0.3. Minix 2.0.3 za to bez problemu uruchamia się na VMWare, VirtualBox, Qemu oraz Hyper-V, ale nie jest w stanie skorzystać z ich interfejsów kart sieciowych. W celu zestawienia połączenia z maszyną Minix należałoby dodać do jądra jego systemu wsparcie dla karty sieciowej emulowanej przez wymienione systemy do wirtualizacji. W pierwszej kolejności należy jednak zainstalować sam system operacyjny, co opisuję w następnym rozdziale.

## 4. Instalacja Miniksa 2.0.3

Instalacja systemu Minix 2.0.3 przebiega poprzez dyskietki instalacyjne. Pliki instalacyjne można znaleźć na oficjalnej stronie Miniksa 3. Paczkę z plikami instalacyjnymi załączyłem także na płycie CD w katalogu *minix\_install/Intel-2.0.3.tar.gz*. Po pobraniu paczki należy ją wypakować w naszym katalogu roboczym. W tym momencie można przejść do przygotowania dyskietek instalacyjnych oraz maszyny wirtualnej.

### 4.1. Przygotowanie dyskietki instalacyjnej

Instalacja składa się z następujących plików:

- *i386/ROOT.MNX* – obraz dyskietki instalacyjnej systemu
- *i386/USR.MNX* – obraz dyskietki instalacyjnej katalogu */usr*
- *i386/NET.TAZ* – opcjonalne pliki do obsługi sieci
- *i386/USR.TAZ* – pliki binarne katalogu */usr*
- *src/SYS.TAZ* – kod źródłowy systemu
- *src/CMD.TAZ* – kod źródłowy poleceń

W celu instalacji systemu należy w pierwszej kolejności stworzyć obraz dyskietki instalacyjnej systemu. Pliki *ROOT.MNT* oraz *USR.MNX* są plikami potrzebnymi do minimalnej instalacji systemu, po której można zainstalować resztę systemu. Plik *ROOT.MNX* waży 480KB, a plik *USR.MNX* waży 720KB, co oznacza, że razem nie przekraczają pojemności dyskietki o pojemności 1440KB i mogą zostać scalone do jednej dyskietki. Korzystając z systemu operacyjnego z rodziny UNIX, w katalogu roboczym, w którym wypakowaliśmy pliki instalacyjne Miniksa 2.0.3 tworzymy pusty obraz dyskietki za pomocą komendy:

```
dd if=/dev/zero of=boot.flp bs=1k count 1440
```

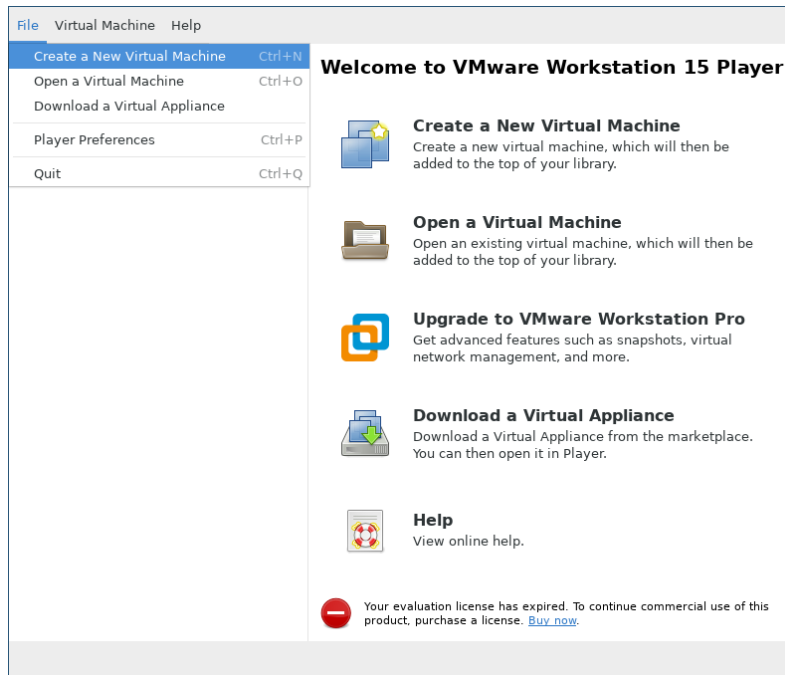
Komenda ta stworzy plik *boot.flp* wypełniony zerami o wielkości 1440KB. W następnym kroku wypełniamy obszar dyskietki zawartością plików *ROOT.MNX* oraz *USR.MNX*:

```
cat Intel-2.0.3/i386/ROOT.MNX Intel-2.0.3/i386/USR.MNX | \  
dd of=boot.flp conv=notrunc
```

Wywołanie komendy wypełni zawartość pliku będącego obrazem dyskietki plikami instalacyjnymi Miniksa. Przygotowany plik *boot.flp* załączyłem także na płycie CD w katalogu *minix\_install/boot.flp*. Mając już medium instalacyjne, możemy przejść do skonfigurowania maszyny wirtualnej w środowisku VMWare.

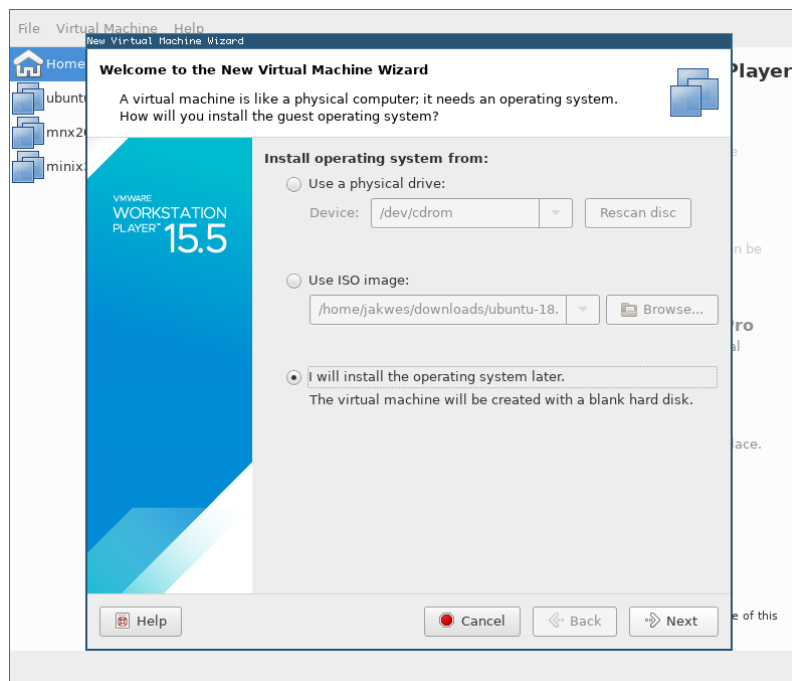
### 4.2. Przygotowanie maszyny wirtualnej w środowisku VMWare

Uruchamiamy oprogramowanie VMWare Player. Z górnego menu w oknie programu wybieramy *File > Create a New Virtual Machine*.



Rysunek 1: Vmware Player – tworzenie nowej maszyny wirtualnej

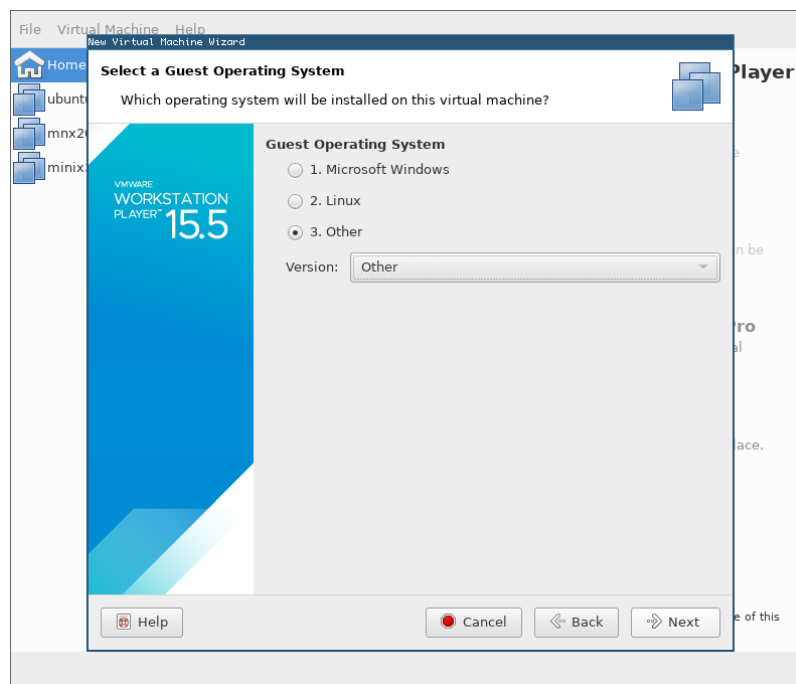
W pierwszej kolejności zostaniemy zapytani o medium instalacyjne systemu. Wybieramy opcję *I will install operating system later* i wciskamy przycisk *Next*.



Rysunek 2: VMWare Player – pytanie o medium instalacyjne systemu

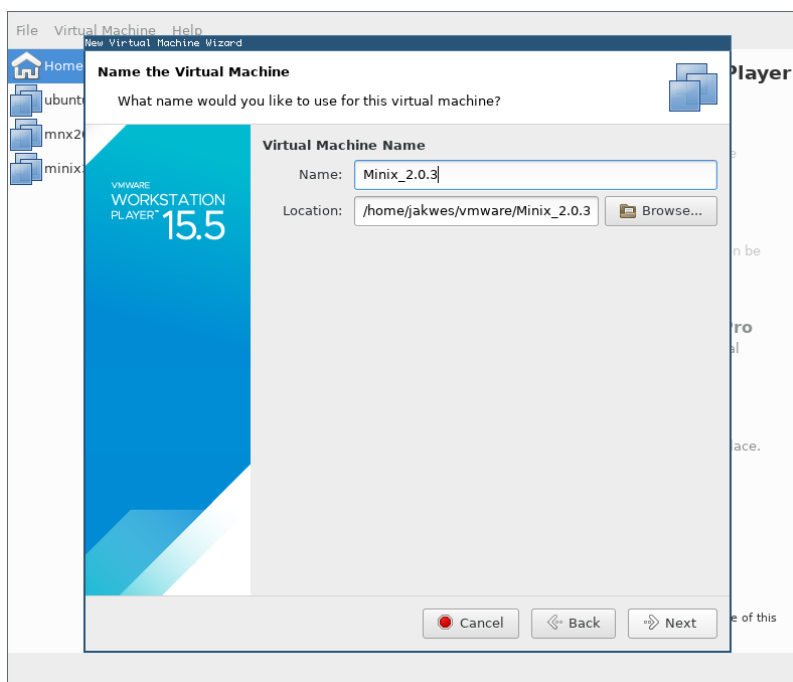
Następnie zapytani o rodzaj instalowanego systemu operacyjnego wybieramy *Other > Other*.





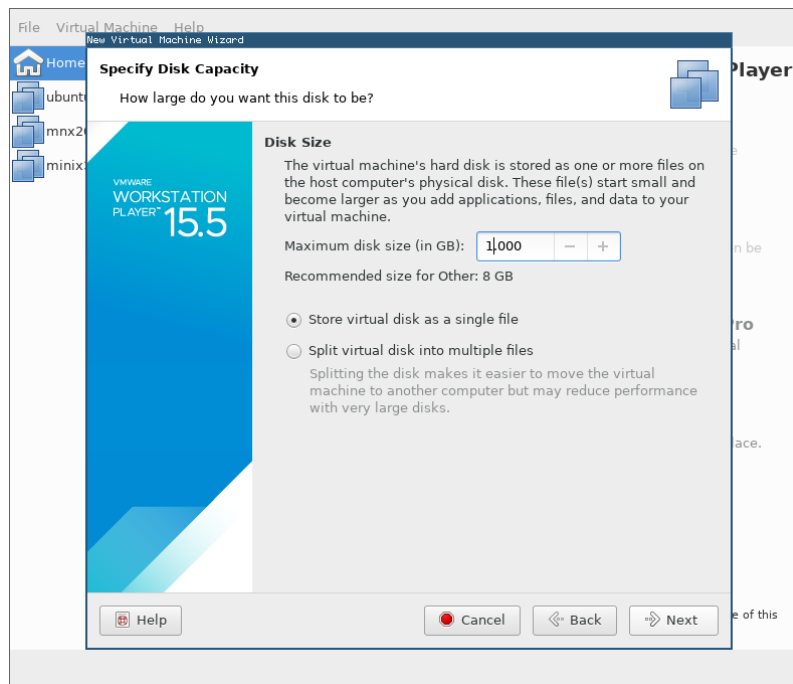
Rysunek 3: VMWare Player – wybór typu systemu operacyjnego

Teraz możemy nadać nazwę naszej maszynie wirtualnej. Jeżeli istnieje taka potrzeba, to można także zmienić katalog, w którym znajdzie się maszyna.



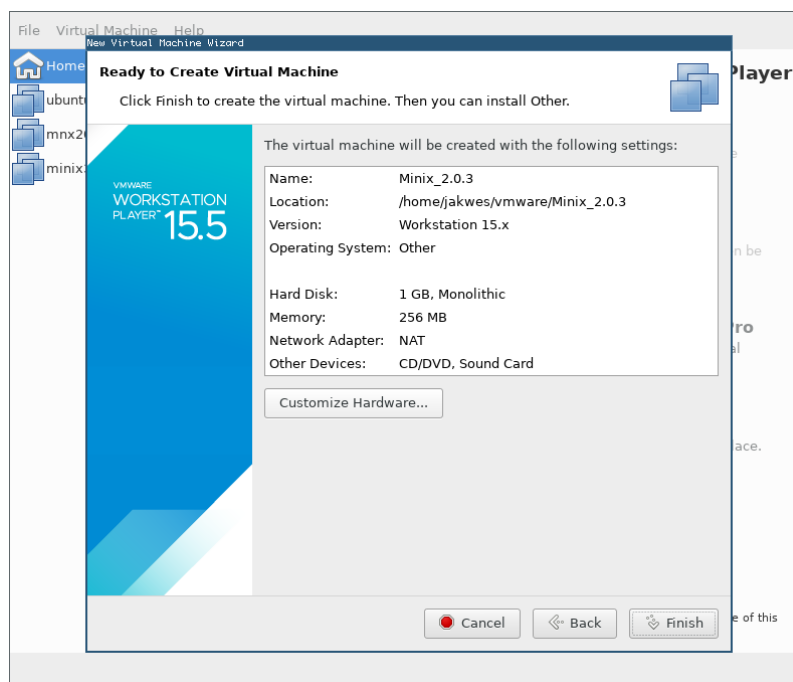
Rysunek 4: VMWare Player – wybór nazwy maszyny wirtualnej

W kolejnym kroku możemy wybrać wielkość dysku wirtualnego oraz metodę jego przechowywania. Podana wielkość jest wielkością maksymalną, plik będzie zajmował mniej miejsca na dysku, jeżeli nie jest zapełniony, dlatego też dobrze jest wybrać wielkość z zapasem. Wybieramy wartość *Maximum disk size (in GB):* równą 1.000 oraz zaznaczamy *Store virtual disk as a single file.*



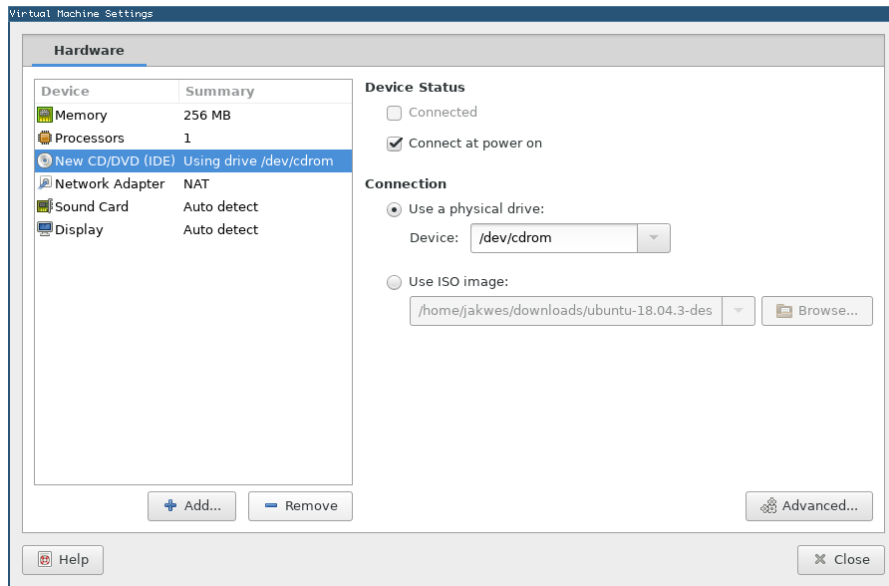
Rysunek 5: VMWare Player – konfiguracja dysku wirtualnego maszyny

Ukaże nam się podsumowanie maszyny wirtualnej. Przed zakończeniem wciskamy przycisk *Customize Hardware...*, aby skonfigurować urządzenia podłączone do maszyny.



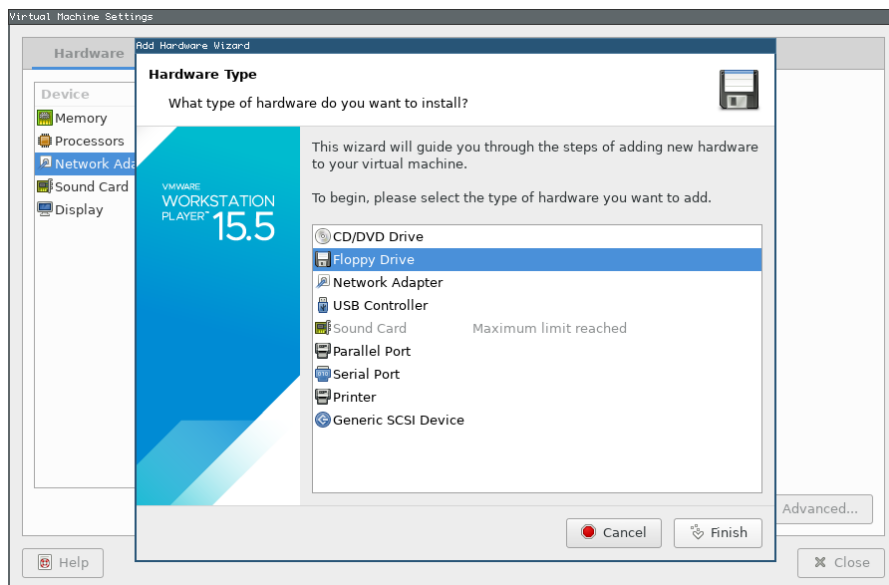
Rysunek 6: VMWare Player – podsumowanie

Na naszej maszynie wirtualnej nie będziemy korzystać z interfejsu czytnika płyt CD/DVD, dlatego wybieramy urządzenie *New CD/DVD (IDE)* i wciskamy przycisk *Remove*.



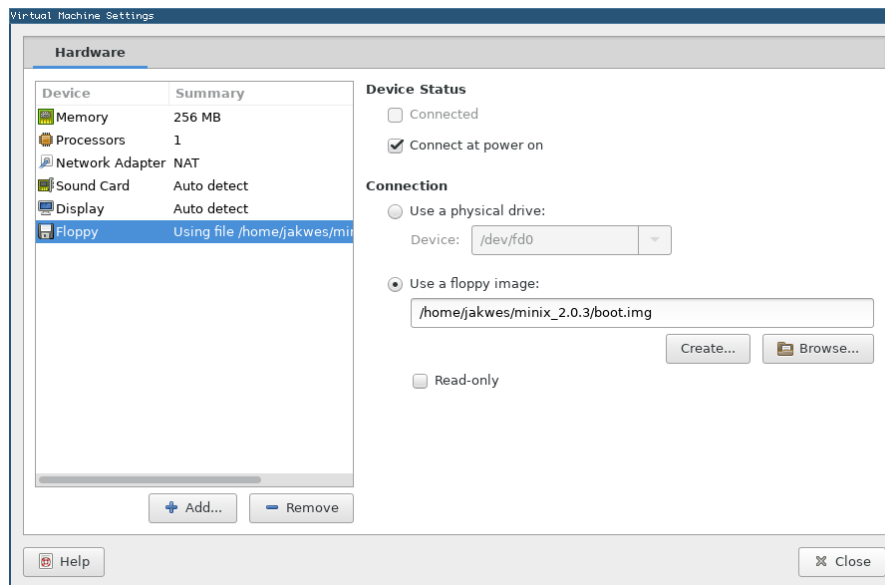
Rysunek 7: VMWare Player – ekran konfiguracji sprzętu maszyny wirtualnej

Do instalacji będziemy potrzebować czytnika dyskietek. Wybieramy przycisk *Add...* i z dostępnych opcji wybieramy *Floppy Drive* i wciskamy *Finish*.



Rysunek 8: VMWare Player – dodawanie czytnika dyskietek do maszyny

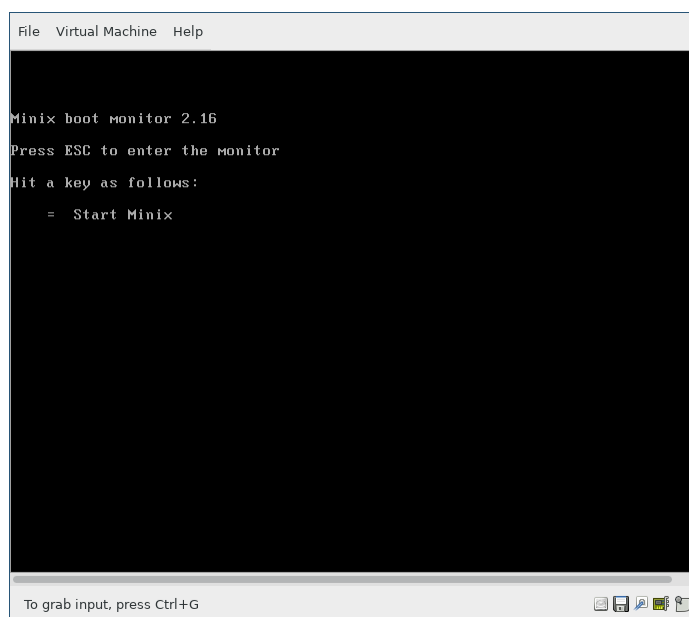
Po dodaniu czytnika zaznaczamy go, w kategorii *Connection* wibieramy opcję *Use a floppy image:* i wciskamy przycisk *Browse...*, aby dodać obraz dyskietki *boot.flp*, który wytworzyliśmy wcześniej. Po wybraniu dyskietki instalacyjnej możemy zamknąć okno konfiguracji przyciskiem *Close*. Maszyna jest gotowa do uruchomienia i instalacji systemu operacyjnego Minix.



Rysunek 9: VMWare – wybór obrazu dyskiety instalacyjnej

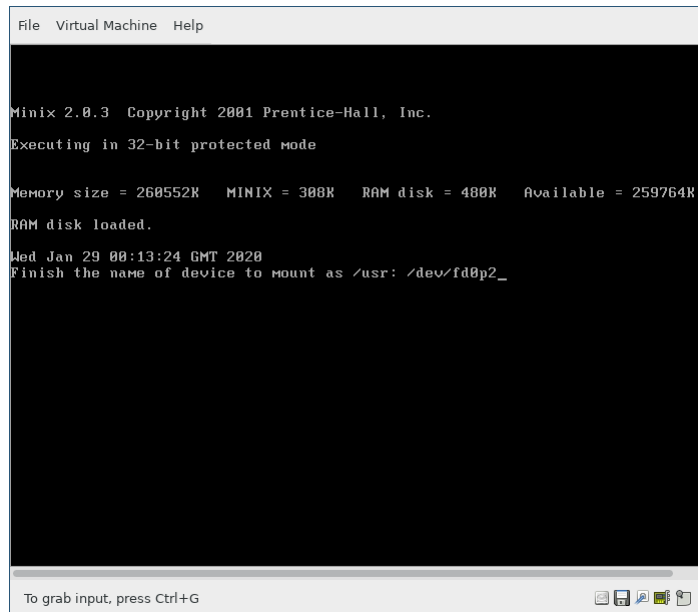
### 4.3. Instalacja systemu operacyjnego Minix 2.0.3

Teraz możemy uruchomić maszynę wirtualną. Na ekranie powinniśmy zobaczyć monitor inicjacji systemu. Wciskamy klawisz „=” na klawiaturze, aby uruchomić system.



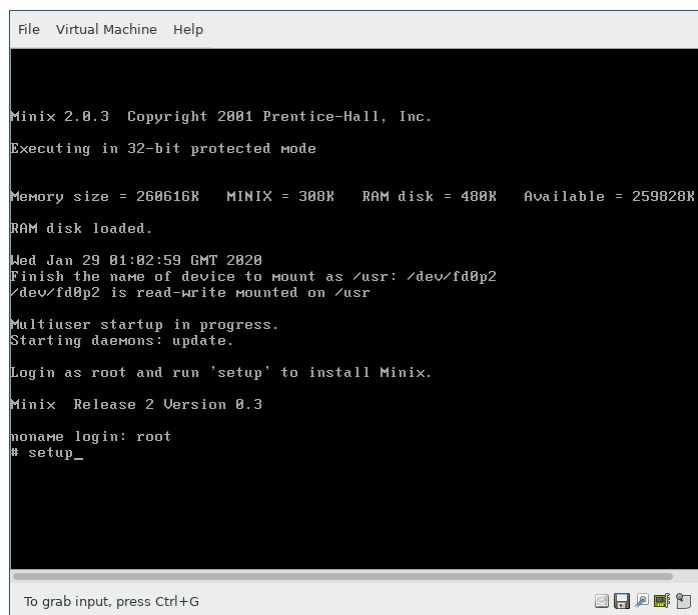
Rysunek 10: Minix – monitor inicjacji systemu Minix

System Minix zacznie się wczytywać z dyskiety. Po uruchomieniu system zapyta się nas o nazwę urządzenia, na którym znajdują się pliki katalogu `/usr`. Pliki te znajdują się na partycji nr. 2 naszej dyskiety, dlatego wpisujemy `fd0p2` (dyskieta 0, partycja 2) i wciskamy klawisz `Enter`.



Rysunek 11: Minix – wybór urządzenia z plikami katalogu /usr

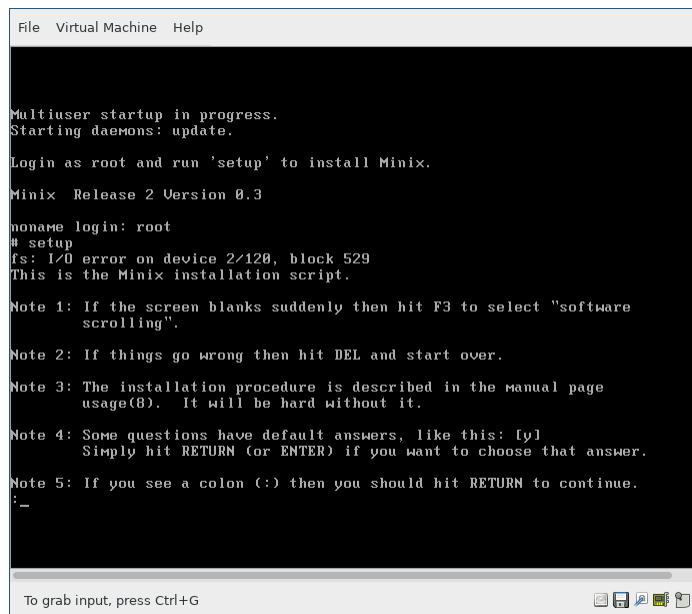
Powinniśmy ujrzeć ekran logowania się do systemu. Zgodnie z instrukcjami na ekranie logujemy się jako użytkownik *root* i w celu instalacji systemu wywołujemy polecenie *setup*.



Rysunek 12: Minix – ekran logowania

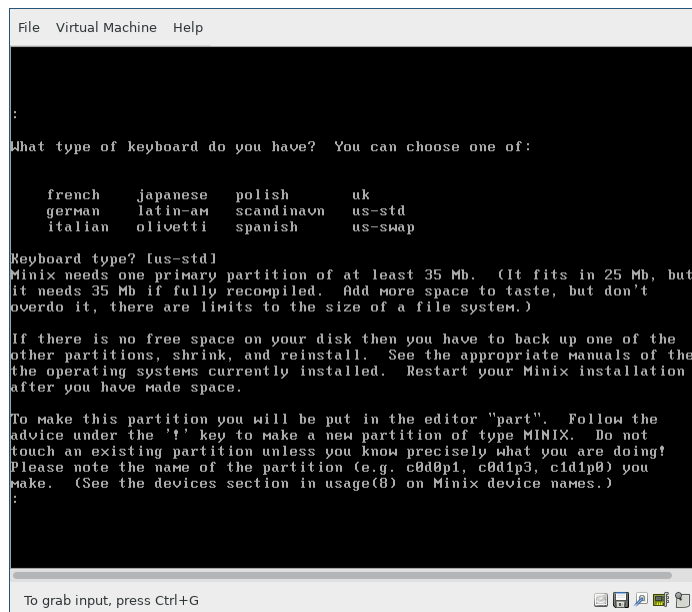
Dokładnie czytamy instrukcję instalacji systemu. Po przeczytaniu wciskamy klawisz *Enter*.

Wiadomościami „*fs: I/O error on device 2/120*” nie należy się przejmować, chyba, że pojawiają się one jeszcze w trakcie montowania katalogu */usr*. Jeśli uniemożliwiają one uruchomienie systemu, należy zajrzeć do rozdziału 4.5., w którym opisują rozwiązanie problemu.



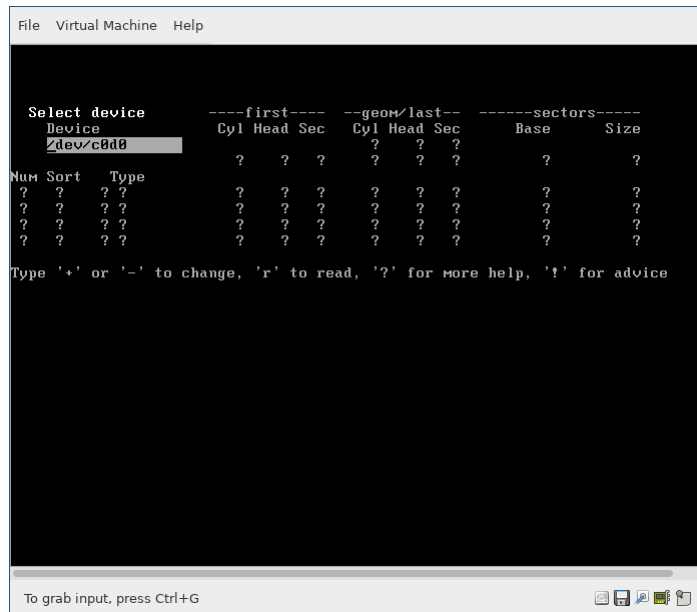
Rysunek 13: Minix – instrukcje do instalacji systemu Minix

W pierwszej kolejności zostaniemy zapytani o typ posiadanej klawiatury. Na tym etapie możemy wybrać domyślną klawiaturę *us-std* wciskając klawisz *Enter*. Następnym krokiem jest podział partycji dysku wirtualnego. Należy zapoznać się z instrukcją i nacisnąć klawisz *Enter*, przeniesie to nas do programu *part* służącego po podziału partycji dysku.



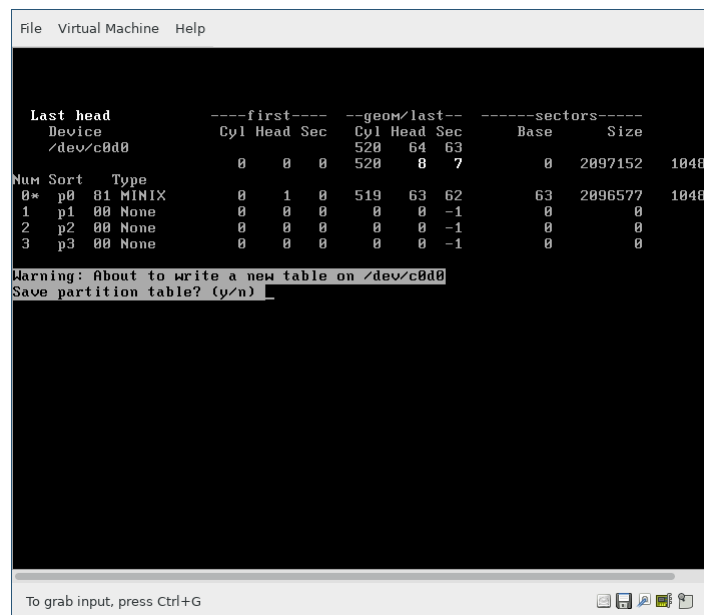
Rysunek 14: Minix – instrukcja partycjonowania dysku

Program *part* jest programem graficznym, należy zapoznać się z jego obsługą wciskając klawisz „?” na klawiaturze. W pierwszej kolejności musimy odczytać tablicę partycji na naszym dysku wirtualnym, który powinien być identyfikowany przez system Minix jako */dev/c0d0*. Naprowadzając kursor za pomocą strzałek na klawiaturze na pole znajdujące się pod *Device* upewniamy się, że wypisuje ono */dev/c0d0*. Jeśli nie, to możemy zmieniać jego wartość za pomocą klawiszy „+” i „-”.



Rysunek 15: Minix – interfejs programu part

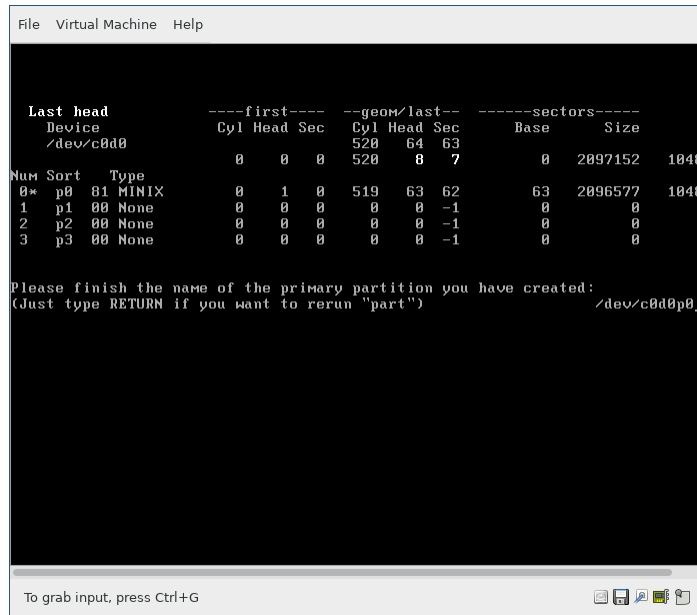
Wciskamy klawisz „r” na klawiaturze w celu odczytania tablicy partycji. Pola wcześniej wypisujące tylko znak „?” powinny zostać wypełnione liczbami. W celu zainstalowania systemu będziemy potrzebowali jednej partycji typu *MINIX*, którą następnie instalator sam podzieli na trzy podstawowe partycje. W tym celu naprowadzamy kursor na liczbę 00 pod kolumną *Type* wiersza partycji p0 i za pomocą klawiatury wpisujemy numer 81. Nazwa obok liczby powinna zmienić się z *None* na *MINIX*. Możemy także naprowadzić kursor na nazwę typu partycji, a następnie wybrać typ *MINIX* za pomocą klawiszy „+” i „-”. W kolejnych kolumnach ustawiamy położenie partycji na dysku. Pozycje możemy wpisywać ręcznie albo przeglądać zalecane miejsca za pomocą klawisza „m”. Zalecana konfiguracja przedstawiona jest na następnej ilustracji.



Rysunek 16: Minix – wypełnienie tabeli partycji dysku

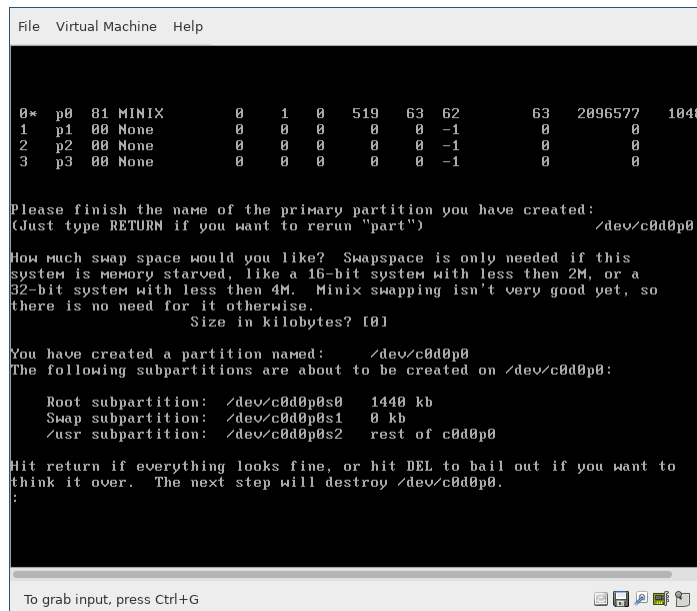
Po wypełnieniu tabeli wciskamy klawisz „w”, aby zapisać tabelę partycji oraz potwierdzamy zapis wpisując „y” i wciskając klawisz *Enter*. Wciskając klawisz „q” zamykamy program *part*.

Po wyjściu z programu instalator systemu powinien zapytać nas o nazwę partycji, którą właśnie utworzyliśmy. Wpisujemy `/dev/c0d0p0` i wciskamy klawisz *Enter*.



Rysunek 17: Minix – wybór stworzonej partycji

Następnie zostaniemy zapytani o wielkość partycji swap. Jako że Minix na dzisiejsze standardy potrzebuje bardzo mało pamięci operacyjnej, oraz że pamięć tą zawsze możemy rozszerzyć w konfiguracji maszyny wirtualnej, zostawiamy domyślną wartość `0` wciskając klawisz *Enter*. Powinniśmy ujrzeć podsumowanie podziału naszej partycji `/dev/c0d0p0` na trzy podpartycje. Jeżeli wszystko się zgadza wciskamy klawisz *Enter*.



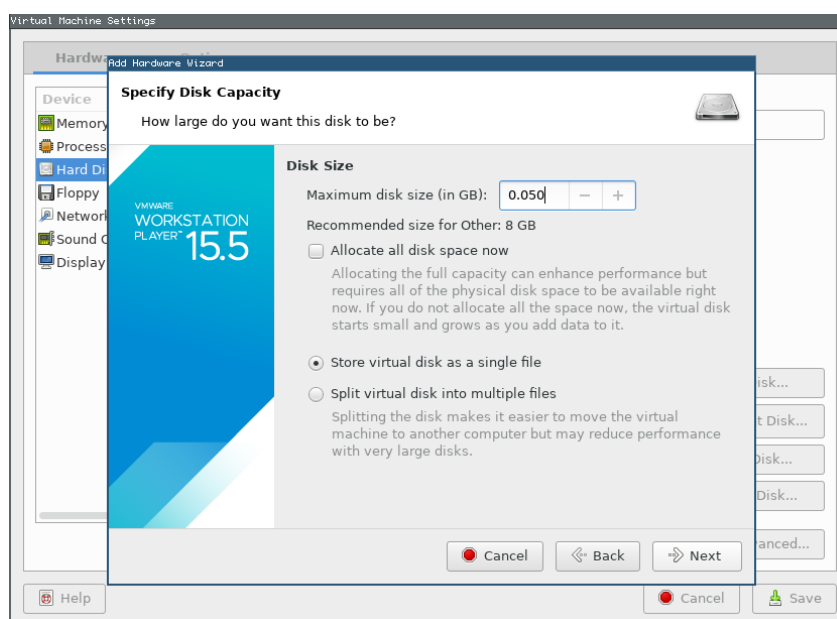
Rysunek 18: Minix – podsumowanie podziału partycji

Minix przystąpi do formatowania dysku wirtualnego, podziału partycji oraz do przekopiowania plików z dyskietki na dysk. Po zakończeniu powinniśmy zostać poinformowani o zakończeniu. Wykonujemy polecenie `halt` i z poziomu VMWare Player wybieramy opcje *Virtual Machine* >



*Power > Power Off Guest*, aby zamknąć maszynę wirtualną. Na tym etapie zainstalowane zostały podstawowe pliki systemu Minix znajdujące się na dyskiecie instalacyjnej. W następnym etapie zainstalujemy resztę plików katalogu */usr*, pliki do obsługi sieciowej, kod źródłowy systemu oraz kod źródłowy poleceń. Elementy te można podobnie jak bazowy system zainstalować poprzez dyski, ale do pełnej instalacji wszystkich pakietów potrzebowalibyśmy aż dziewięciu obrazów dyski (trzy na *USR.TAZ*, jedną na *NET.TAZ*, dwie na *SYS.TAZ* oraz trzy na *CMD.TAZ*). Jako, że mamy już zainstalowaną bazę systemu Minix, zamiast z dyski możemy skorzystać z drugiego dysku wirtualnego, którego użyjemy jako nośnika, na którym przeniesiemy resztę plików instalacyjnych na system operacyjny.

W oprogramowaniu VMWare Player przechodzimy do konfiguracji maszyny wirtualnej z Miniksem. Za pomocą opcji *Add...* dodajemy drugi dysk twardy do maszyny, jako typ dysku zaznaczamy *IDE*, następnie wybieramy *Create a new virtual disk* oraz wybieramy rozmiar dla dysku. Proces ten odbywa się podobnie jak dla pierwszego dysku tworzonego razem z maszyną.



Rysunek 19: VMWare Player – tworzenie drugiego dysku wirtualnego

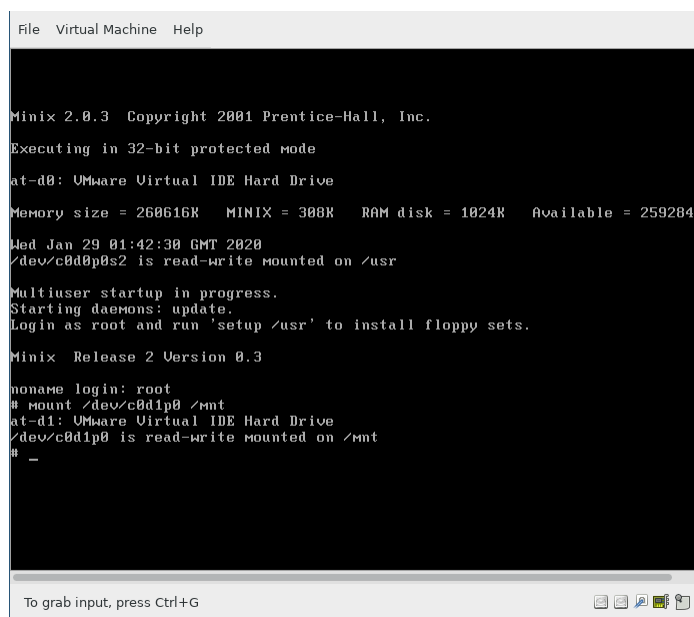
Na końcu wybieramy nazwę dla dysku, ja nazwałem go *Minix\_2.0.3\_installation\_files.vmdk*. Po zapisaniu konfiguracji maszyny przyciskiem *Save* możemy ponownie uruchomić maszynę wirtualną. Uruchamiamy system Minix i logujemy się jako użytkownik *root*. Następnie sformatujemy nasz drugi dysk, aby mógł być rozpoznawalny przez Miniksa oraz abyśmy mogli kopiować na niego pliki. Za pomocą narzędzia *part* tworzymy tablicę partycji dla drugiego dysku analogicznie do dysku pierwszego. Ustawiamy typ *MINIX* oraz wielkość dla partycji */dev/c0d1p0*. Po zapisaniu zmian musimy jeszcze stworzyć system plików na tej partycji. Wychodzimy z programu *part* i wywołujemy komendę *mkfs /dev/c0d1p0 50000*, gdzie ostatni argument to wielkość systemu plików wyrażona w KB. Partycja stworzona w ten sposób powinna dać się montować i powinna pozwalać na zapis plików. Możemy przetestować montowanie dysku na Miniksie za pomocą komendy *mount /dev/c0d1p0 /mnt*. W katalogu */mnt* powinniśmy móc tworzyć i edytować pliki. Zamykamy system za pomocą polecenia *shutdown*, po powrocie do monitora inicjacji systemu możemy zamknąć maszynę wirtualną z poziomu oprogramowania VMWare Player.

W kolejnym etapie będziemy chcieli wypełnić dysk wirtualny niezbędnymi plikami instalacyjnymi. Można to zrobić na wiele sposobów. Jeżeli system zainstalowany na twoim komputerze pozwala

na zamontowanie pliku dysku wirtualnego w formacie VMDK możesz to zrobić w ten sposób. Uniwersalnym sposobem jest skonfigurowanie maszyny wirtualnej z systemem Ubuntu. Po instalacji systemu Ubuntu w konfiguracji maszyny wirtualnej dodajemy istniejący dysk *Minix\_2.0.3\_installation\_files.vmdk* jako drugi dysk maszyny. Po uruchomieniu system Ubuntu powinien wykryć dysk oraz pozwolić na jego zamontowanie. Mając dostęp do sieci na systemie Ubuntu możemy pobrać pliki instalacyjne Miniksa 2.0.3 oraz przekopiować je na dysk *Minix\_2.0.3\_installation\_files.vmdk*. Pliki, które nas interesują to:

- `i386/NET.TAZ` – pliki do obsługi sieci
- `i386/USR.TAZ` – pliki binarne katalogu `/usr`
- `src/SYS.TAZ` – kod źródłowy systemu
- `src/CMD.TAZ` – kod źródłowy poleceń

Po przekopiowaniu plików na dysk zamykamy system Ubuntu. Uruchamiamy maszynę wirtualną z Miniksem, logujemy się jako użytkownik *root*. Za pomocą polecenia `mount /dev/c0d1p0/mnt` montujemy dysk w katalogu `/mnt`.

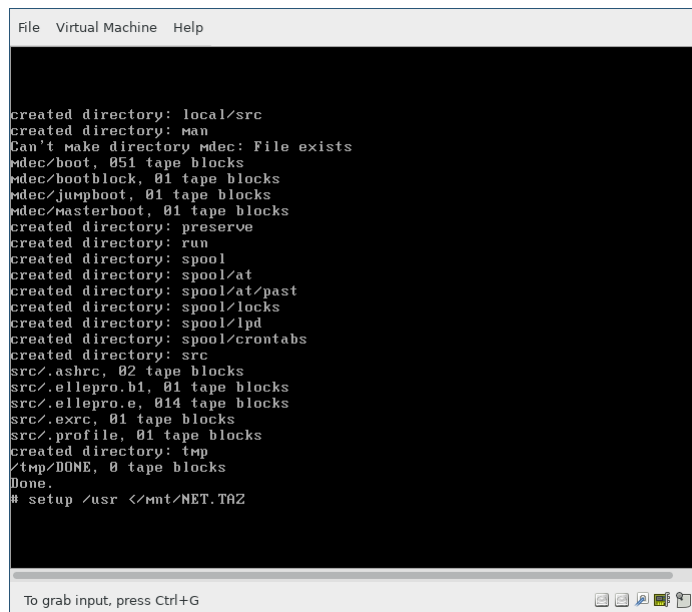


```
File Virtual Machine Help
Minix 2.0.3 Copyright 2001 Prentice-Hall, Inc.
Executing in 32-bit protected mode
at-d0: VMware Virtual IDE Hard Drive
Memory size = 260616K MINIX = 300K RAM disk = 1024K Available = 259204K
Wed Jan 29 01:42:30 GMT 2020
/dev/c0d0p0s2 is read-write mounted on /usr
Multiuser startup in progress.
Starting daemons: update.
Login as root and run 'setup /usr' to install floppy sets.
Minix Release 2 Version 0.3
noname login: root
# mount /dev/c0d1p0 /mnt
at-d1: VMware Virtual IDE Hard Drive
/dev/c0d1p0 is read-write mounted on /mnt
# _
To grab input, press Ctrl+G
```

Rysunek 20: Minix – montowanie dysku z plikami instalacyjnymi

Uruchamiając polecenie `ls /mnt` powinniśmy zobaczyć wylistowane pliki, które przekopiowaliśmy na dysk z poziomu maszyny Ubuntu. W celu instalacji reszty systemu operacyjnego wywołujemy polecenie `setup` z odpowiednimi parametrami:

```
setup /usr </mnt/USR.TAZ
setup /usr </mnt/NET.TAZ
setup /usr </mnt/SYS.TAZ
setup /usr </mnt/CMD.TAZ
```



```
File Virtual Machine Help
created directory: local/src
created directory: man
Can't make directory mdec: File exists
mdec/boot, 051 tape blocks
mdec/bootblock, 01 tape blocks
mdec/jumpboot, 01 tape blocks
mdec/masterboot, 01 tape blocks
created directory: preserve
created directory: run
created directory: spool
created directory: spool/at
created directory: spool/at/past
created directory: spool/locks
created directory: spool/lpd
created directory: spool/crontabs
created directory: src
src/.ashrc, 02 tape blocks
src/.ellepro.b1, 01 tape blocks
src/.ellepro.e, 014 tape blocks
src/.exrc, 01 tape blocks
src/.profile, 01 tape blocks
created directory: tmp
/tmp/DONE, 0 tape blocks
Done.
# setup /usr </mnt/NET.TAZ
```

Rysunek 21: Minix – instalacja plików odpowiedzialnych za obsługę sieci

Po wykonaniu powyższych czterech poleceń system operacyjny powinien zostać zainstalowany w pełni.

Przygotowany plik dysku wirtualnego *Minix\_2.0.3\_installation\_files.vmdk* zawierającego potrzebne pliki instalacyjne załączyłem także na płycie CD w katalogu *minix\_install/Minix\_2.0.3\_installation\_files.vmdk*. Dysk ten może okazać się przydatny także w przyszłości. Do póki maszyna z Miniksem nie ma połączenia sieciowego z innymi maszynami dysk wirtualny może zostać użyty do przenoszenia dodatkowego oprogramowania i plików na maszynę Minix.

Mając gotową maszynę wirtualną z Miniksem mogłem przystąpić do konfiguracji komunikacji sieciowej na niej. Proces ten opisuję w następnym rozdziale.

#### 4.4. Instalacja Miniksa 2.0.3 w środowiskach innych niż VMWare

Rozdział 4.2. opisuje proces instalacji systemu Minix 2.0.3 w środowisku VMWare, ale może także posłużyć za instrukcję do instalacji systemu w innych systemach do wirtualizacji. W wybranym przez siebie środowisku należy stworzyć i skonfigurować maszynę wirtualną w analogiczny sposób do tego opisanego w rozdziale 4.2. dla oprogramowania VMWare. Do instalacji na dowolnym środowisku powinno wystarczyć stworzenie dysku wirtualnego oraz zamontowanie dyskietki instalacyjnej w interfejsie czytnika. Instalacja samego systemu opisana w rozdziale 4.3. jest taka sama dla każdego ze środowisk.

Warto wspomnieć, że oprogramowanie VirtualBox oraz Qemu zawierają wsparcie dla dysków wirtualnych w formacie VMDK. Oznacza to, że dysk, na którym zainstalowaliśmy Miniksa może zostać także uruchomiony na innych środowiskach. Konfiguracja maszyny wirtualnej i dodanie gotowego już dysku z zainstalowanym systemem (w mojej instrukcji jest to *Minix\_2.0.3.vmdk*) powinna wystarczyć do uruchomienia systemu na oprogramowaniu VirtualBox oraz Qemu. Oprogramowanie Hyper-V nie posiada obsługi dysków w formacie VMDK, ale istnieje oficjalne narzędzie *Microsoft Virtual Machine Converter* służące do konwersji tego formatu do formatu obsługiwanego przez Hyper-V.

## 4.5. Rozwiązywanie problemów z dyskieta, „fs: I/O error on device”

Podczas instalacji systemu Minix z dyskietki na VMWare można czasem natrafić na komunikaty o błędach wejścia i wyjścia na urządzeniu czytnika dyskietek. Błędy te potrafią uniemożliwić instalację systemu. Trudno jest mi określić ich dokładną przyczynę, może ona leżeć po stronie implementacji emulowanego przez VMWare interfejsu czytnika dyskietek albo po stronie Miniksa nie będącego w stanie poprawnie obsłużyć czytnika. W pierwszej kolejności należy jednak upewnić się, że stworzyliśmy dyskietkę poprawnie według instrukcji w rozdziale 4.1. i stworzyć ją ponownie. Można także użyć przygotowanego już przeze mnie obrazu dyskietki z katalogu *minix\_install/boot.flp*.

Jeżeli stworzenie dyskietki ponownie nie naprawiło problemu należy spróbować zmienić wersję maszyny wirtualnej. Otwieramy program VMWare Workstation. Jeżeli nasza maszyna wirtualna nie widnieje w bibliotece to otwieramy ją ręcznie. Następnie klikamy prawym przyciskiem myszy na naszej maszynie, po czym wybieramy opcję *Manage > Change Hardware Compatibility*. Z listy dostępnych wersji kompatybilności wybieramy *Workstation 5.x*. W ostatnim kroku możemy stworzyć zmodyfikowaną maszynę jako kopię istniejącej, aby nie nadpisywać konfiguracji oryginalnej maszyny. Uruchamiamy maszynę skonfigurowaną pod wersję *Workstation 5.x* i próbujemy ponownie zainstalować system.

Powyższe pomoce powinny rozwiązać problem. Jeżeli problem dalej występuje, zamiast instalować system ręcznie można skorzystać z obrazu dysku wirtualnego ze świeżo zainstalowanym systemem. Plik z dyskiem znajduje się w katalogu *minix\_install/Minix\_2.0.3.vmdk* i jest obrazem dysku po świeżej instalacji systemu.

## 5. Sterownik karty sieciowej AMD Lance

Jak wspominałem w rozdziale 3., Minix 2.0.3 zawiera standardowo sterowniki do obsługi kart sieciowych z rodzin DP8390, Western Digital WD80x3, Novell NE1000/2000 oraz 3Com Etherlink II 3C503. Żaden z wymienionych interfejsów nie jest emulowany przez oprogramowanie VMWare, VirtualBox ani Qemu.

Na oficjalnej stronie Miniksa 2, poza dystrybucją samego systemu operacyjnego, można znaleźć także dodatkowe oprogramowanie przygotowane na ten system przez sympatyków. Wśród tego oprogramowania można znaleźć sterowniki kart sieciowych rodziny AMD Lance oraz Intel Pro przygotowane dla Miniksa 2.0.4. Sterowniki te są konwersją sterowników przygotowanych dla Miniksa 2.0.2 [5]. Przeglądając Internet nie znalazłem konwersji żadnego z wymienionych sterowników dla Miniksa 2.0.3. Jako że emulację sterownika PCNet z rodziny AMD Lance wspierają wszystkie trzy wcześniej wymienione oprogramowania do wirtualizacji (Qemu, VirtualBox, VMWare) sterownik taki pozwoliłby na wykrycie interfejsu emulowanego przez maszynę i umożliwiłby komunikację z innymi maszynami wirtualnymi w tej samej sieci. Jako że istnieją wersje tych sterowników dla Miniksa 2.0.2 oraz 2.0.4, ale nie istnieje konwersja na Miniksa 2.0.3, w pierwszej kolejności postanowiłem skupić się na przetestowaniu działania obu z wersji sterowników.

### 5.1. Test sterownika dla Miniksa 2.0.2

Do przetestowania obu wersji sterowników potrzebne mi były systemy Minix w wersjach 2.0.2 oraz 2.0.4. W pierwszej kolejności zdecydowałem się na przetestowanie oryginalnego sterownika dla wersji 2.0.2. W środowisku VMWare Player przygotowałem dwie maszyny wirtualne: jedną dla systemu operacyjnego Minix 2.0.2, drugą dla systemu operacyjnego Ubuntu. Maszyna wirtualna z Miniksem posłuży mi do przetestowania poprawności działania sterownika, a maszyna Ubuntu zostanie użyta do sprawdzenia połączenia sieciowego z maszyną Minix oraz do przetestowania oprogramowania sieciowego dostępnego na Miniksie. W konfiguracji obu maszyn wirtualnych należało skonfigurować interfejsy obu kart sieciowych tak, aby mogły się ze sobą porozumieć. Przy wybraniu opcji „NAT” oba interfejsy sieciowe znajdują się w jednej sieci NAT, ruterem w tej konfiguracji jest komputer, na którym uruchamiane są maszyny wirtualne. Konfiguracja NAT pozwala na komunikację maszyn wirtualnych ze sobą oraz na zewnątrz do Internetu, ruch od strony komputera jest blokowany. Następnym krokiem była instalacja systemów operacyjnych na obu maszynach. Instalacja systemu Minix 2.0.2 nie różni się od procesu instalacji wersji 2.0.3 opisanej w rozdziale 5. Do przeniesienia plików ze sterownikiem karty sieciowej AMD Lance użyłem drugiego dysku wirtualnego. Przeniesienie plików odbywa się analogicznie do procesu opisanego w rozdziale 5.: dysk zostaje zamontowany na maszynie z Ubuntu, z poziomu systemu Ubuntu pobierana zostaje dystrybucja sterownika oraz kopiowana na dysk służący za nośnik. Następnie dysk ten jest montowany na maszynie z Miniksem.

Proces instalacji sterownika przebiega następująco. Paczkę z dystrybucją na Miniksie kopiujemy do katalogu `/usr`, dekompresujemy narzędziem `uncompress neweth.tar.Z`, a następnie zdekompresowane archiwum wypakowujemy poleceniem `tar xf neweth.tar`. Przy wypakowywaniu plików należy zachować ostrożność, wypakowane pliki nadpiszą pliki znajdujące się na systemie. W przypadku czystej instalacji systemu nie ma żadnych problemów z kompatybilnością z innym niestandardowym oprogramowaniem, więc można te pliki wypakować i nadpisać. Sterownik zostanie skompilowany wraz z jądrem, przed tym należy jednak włączyć obsługę sieci. W pliku `/usr/include/minix/config.h` znajdujemy linijkę:

```
#define ENABLE_NETWORKING 0
```

I zmieniamy wartość definicji *ENABLE\_NETWORKING* na 1:

```
#define ENABLE_NETWORKING 1
```

Warto także zwiększyć ilość pseudoterminali. Pseudoterminale są mechanizmem, z którego korzystają aplikacje umożliwiające zdalne logowanie t.j. rlogin oraz telnet. W tym celu znajdujemy definicję *NR\_PTYS* i nadajemy jej wartość większą od zera, np. 4.

Po zapisaniu zmian w pliku możemy przejść do kompilacji jądra. W katalogu */usr/src/tools* wykonujemy komendę *make hdbboot*. Jądro systemu zostanie zbudowane, a następnie podmienione. Zamykamy system poleceniem *shutdown*. Wracamy w ten sposób do monitora inicjacji systemu. Wciskamy klawisz *Escape* oraz wpisujemy do konsoli:

```
LANCE0=on  
save  
boot
```

*LANCE0* jest zmienną odpowiadającą za włączenie sterownika AMD Lance. Polecenie *save* zapisuje edycję zmiennych monitora inicjacji systemu, a polecenie *boot* uruchamia system.

Przy uruchomieniu system prawidłowo rozpoznał kartę sieciową jako „PCnet/PCI-II 79C970A”. Następnie nadałem maszynie stały adres IP za pomocą polecenia *ifconfig* w tej samej podsieci, co maszyna Ubuntu. Przystąpiłem do testów komunikacji po sieci. Maszyna Minix wykrywa maszynę Ubuntu oraz komputery w sieci Internet. Z poziomu maszyny Ubuntu udało mi się połączyć z serwerami usług RSH oraz FTP uruchomionymi na Miniksie. Testy potwierdziły poprawne działanie sterownika na Miniksie 2.0.2.

Test ten został powtórzony także na oprogramowaniu VirtualBox oraz Qemu. W obu środowiskach sterownik zadziałał prawidłowo i umożliwił komunikację sieciową z maszyną Ubuntu.

## 5.2. Test sterownika dla Miniksa 2.0.4

W następnym kroku zdecydowałem się przetestować także konwersję sterownika dla Miniksa 2.0.4. Proces testowania sterownika przebiegł bardzo podobnie do rozdziału 5.1. Instalacja systemu Minix 2.0.4, edycja pliku */usr/include/minix/config.h* oraz kompilacja jądra jest analogiczna do Miniksa 2.0.2. Inaczej wygląda konfiguracja sterownika. Plik */etc/inet.conf* należy wyedytować, aby jego zawartość wyglądała w następujący sposób:

```
eth0 LANCE 0 { default; };
```

Dodatkowo należy ustawić następujące parametry monitora inicjacji systemu:

```
LANCE0=pci  
servers=inet
```

Po restarcie maszyny przetestowałem i potwierdziłem działanie sterownika.

Test został powtórzony na oprogramowaniu VirtualBox oraz Qemu. W obu środowiskach, podobnie jak w przypadku testu Miniksa 2.0.2, sterownik zadziałał prawidłowo i umożliwił komunikację sieciową z maszyną Ubuntu.

Powyższe testy potwierdziły działanie dwóch wersji sterownika w trzech różnych środowiskach do wirtualizacji. W celu umożliwienia obsługi sieci na Miniksie 2.0.3 zdecydowałem się przekonwertować ten sterownik na Miniksa 2.0.3.

### **5.3. Konwersja sterownika karty sieciowej AMD Lance na Miniksa 2.0.3**

W celu dokonania konwersji sterownika AMD Lance musiałem zapoznać się ze zmianami w jądrze, jakie zaszły między wersją 2.0.2 oraz 2.0.3 Miniksa. Początkowe próby zainstalowania sterownika z Miniksa 2.0.2 lub 2.0.4 na Miniksie 2.0.3 skończyły się porażką i popłochem jądra (ang. kernel panic). Zarówno sterownik dla wersji 2.0.2, jak i jego konwersja dla wersji 2.0.4 opiera się na sterowniku karty sieciowej DP8390 dostępnej w standardowej dystrybucji Miniksa, dlatego zacząłem analizować zmiany w tym sterowniku między wersjami Miniksa 2.0.2 i 2.0.3. W kodzie jądra dokonano kilku niewielkich zmian oraz refaktoryzacji. Zmieniono między innymi obsługę pamięci współdzielonej procesu oraz wyszukiwania adresu fizycznego urządzenia do wykonywania operacji wejścia i wyjścia. Na podstawie zmian dokonanych w kodzie sterownika DP8390 przepisałem kod sterownika AMD Lance z wersji Miniksa 2.0.2 na 2.0.3. Po dokonaniu konwersji przetestowałem rozwiązanie w środowisku maszyn wirtualnych VMWare, VirtualBox oraz Qemu. Skonfigurowałem dwie maszyny wirtualne wraz z interfejsem sieciowym PCNet w każdym z środowisk, Po jednej z systemem Minix 2.0.3 ze skonwertowanym sterownikiem AMD Lance oraz Po jednej z systemem Ubuntu. We wszystkich trzech przypadkach system Minix prawidłowo rozpoznaje kartę sieciową oraz pozwala na komunikację w sieci. Z poziomu maszyny Miniksa jesteśmy w stanie połączyć się z maszyną Ubuntu oraz w drugą stronę. Działają programy takie jak RSH oraz FTP. Oznacza to, że podstawowy cel pracy został wykonany.

Wynik konwersji sterownika zawarłem na płycie CD w katalogu *kernel\_ports*. W katalogu *distributions/disk\_images* znajduje się obraz dysku wirtualnego maszyny zawierający zainstalowany skonwertowany sterownik oraz resztę prac wykonanych w następnych rozdziałach. W następnym etapie zdecydowałem się na rozwinięcie dostępnego oprogramowania na Miniksa 2.0.3, jako że jego wybór na tym systemie jest bardzo ograniczony. Posiadam już Miniksa z działającą siecią, rozwój dodatkowego oprogramowania jest rzeczą, która mogłaby okazać się użyteczna dla studenta. Jednocześnie nie jest to rzecz krytyczna w kontekście całej pracy. Proces rozwoju samej dystrybucji Miniksa 2.0.3 przygotowanej przeze mnie oraz całego środowiska do pracy na laboratorium może być kontynuowany w przyszłości przez inne osoby chętne do pomocy i zainteresowane tą tematyką.





## 6. Oprogramowanie SSH na Miniksa 2.0.3

Po przekonwertowaniu sterownika karty sieciowej wraz z promotorem zaczęliśmy się zastanawiać jakie dodatkowe oprogramowanie można by przygotować na Miniksa. Wybór jest duży, ale jako że Minix posiada bardzo przestarzałe programy do komunikacji sieciowej (Telnet, RSH) zdecydowaliśmy się na próbę konwersji oprogramowania SSH na Miniksa 2.0.3. SSH jest standardowym narzędziem do komunikacji po sieci dostępnym praktycznie na każdej nowoczesnej dystrybucji zapewniającym szyfrowany transfer danych [10]. O ile szyfrowany transfer danych nie jest w moim rozwiązaniu wymagany, gdyż interfejs sieciowy maszyny wirtualnej z Miniksem nie będzie dostępny poza lokalną siecią komputera, to jest on narzędziem, do którego większość studentów jest przyzwyczajona oraz jest w jego obsłudze rozeznana. Konwersja SSH pozwoli także na rozeznanie się, czy takie aplikacje sieciowe da się przekonwertować na przestarzały system Minix 2.0.3 oraz ewentualnie pozwoli na stworzenie podwalin do konwersji następnych aplikacji sieciowych. Wiele aktualnych dystrybucji systemów opartych o jądro Linuxa posiada klienta SSH zainstalowanego domyślnie. Przestarzałe programy takie jak telnet oraz RSH nie są już standardowo dostarczane wraz z dystrybucją systemu, należy ich szukać w repozytoriach danej dystrybucji.

### 6.1. OpenSSH 1.2.3

Jako, że system Minix 2.0.3 jest bardzo przestarzały wiele z zależności wymaganych przez OpenSSH może być na Miniksie niedostępnych, w szczególności zależności od jądra systemu. Zacząłem więc próbę przekonwertowania oprogramowania OpenSSH od wersji 1.2.3p1 z 2000 roku.

OpenSSH rozwijane jest w pierwszej kolejności na system OpenBSD, część kodu OpenSSH jest zależna od funkcji, zmiennych, struktur i innych elementów występujących w jądrze systemu OpenBSD. W celu przekonwertowania programu OpenSSH na systemy operacyjne z rodziny Unix/Linux inne niż OpenBSD deweloperzy wypuszczają także przenośne wersje systemu zakończone literą „p” i numerem opublikowanej wersji [12]. Wersja 1.2.3p1 jest pierwszą publikacją przenośnej konwersji wersji 1.2.3. Deweloperzy OpenSSH tworzą wersje przenośne poprzez wyodrębnienie części z unikalnego kodu systemu OpenBSD i rozpowszechnienie jej wraz z dystrybucją OpenSSH. Wersja przenośna OpenSSH jest spakowana razem z potrzebnym kodem OpenBSD, który w razie potrzeby jest kompilowany w trakcie budowania aplikacji. Do konfiguracji kompilacji OpenSSH w wersji przenośnej korzysta z narzędzia GNU Autoconf [12]. Nie wszystkie funkcje jądra systemu OpenBSD da się przenieść na pozostałe platformy w pełnej ich formie, dlatego też zaimplementowane funkcje z OpenBSD są wybrakowane, ale funkcjonalne na tyle, aby OpenSSH działało w pełni prawidłowo. Implementacje te nie powinny być zatem używane w innych aplikacjach sieciowych, gdyż nie wspierają wszystkich funkcjonalności oryginalnych funkcji OpenBSD, ale są one wystarczające do działania OpenSSH.

Poza kodem z OpenBSD OpenSSH jest zależne także od bibliotek OpenSSL oraz Zlib. OpenSSL zapewnia implementacje protokołów kryptograficznych, a Zlib zapewnia mechanizmy kompresji danych. W repozytorium oprogramowania dla Miniksa 2 znajdują się konwersje zarówno biblioteki OpenSSL jak i Zlib na Miniksa. Konwersja OpenSSL w wersji 0.9.6l została przetestowana przez jej autora na Miniksie 2.0.2 i 2.0.4, ale bez problemu kompiluje się i działa także na Miniksie 2.0.3, bez wymagania żadnych poprawek czy zmian w kodzie. Konwersja biblioteki Zlib 1.2.3 także kompiluje się bez problemu na maszynie docelowej. Obie wersje bibliotek spełniają wymagania OpenSSH. Po zainstalowaniu obu bibliotek na maszynie przystąpiłem do kompilacji samego programu.

Pierwszym problemem na jaki napotkałem się podczas konwersji OpenSSH na Miniksa były długości nazw plików. Minix korzysta z własnego systemu plików, który obsługuje pliki o nazwach

długości do 14 znaków. Popularne systemy plików używane na OpenBSD oraz na systemach z rodziny Linux w czasach powstania tej wersji OpenSSH nie posiadały takiego ograniczenia. Zbyt długie nazwy plików źródłowych powodują, że po wypakowaniu archiwum z kodem nazwy te są skracane do 14 pierwszych znaków. Wszelkie odniesienia w kodzie do plików odwołują się do ich pełnych nazw, kiedy to pliki w systemie posiadają nazwy skrócone. Automatyczne skracanie nazw przez system plików powoduje także nadpisywanie się plików, w szczególności dla plików źródłowych i nagłówkowych języka C. Rozszerzenia są częścią nazwy pliku, więc skrócenie ich do pierwszych 14 znaków powoduje kolizje w nazwach. By rozwiązać ten problem musiałem masowo poskracać nazwy plików tak, aby nie występowały między nimi kolizje. Wszelkie odniesienia do nazw plików także musiały zostać pozamieniane na ich skrócone odpowiedniki. Z pomocą narzędzi do masowego wyszukiwania i zamieniania fraz przygotowałem dystrybucję OpenSSH ze skróconymi do maksymalnie 14 znaków nazwami plików na komputerze z systemem Ubuntu, następnie gotową dystrybucję przenieśliem na maszynę z Miniksem za pośrednictwem sieci poprzez protokół FTP. Dzięki temu mogłem już zacząć próby kompilacji aplikacji narzędziem Make.

Kolejnym napotkanym problemem były brakujące funkcjonalności bibliotek oraz jądra systemu Minix. O ile wersja przenośna OpenSSH zawiera w swojej dystrybucji część potrzebnego kodu z OpenBSD, to nadal zakłada istnienie w systemie takich funkcjonalności jak gniazda BSD do komunikacji sieciowej. W przeciwieństwie do Linuxa system Minix posiada własny interfejs gniazd sieciowych niekompatybilny z gniazdami BSD. Na szczęście w repozytorium oprogramowania Miniksa 2 można znaleźć bibliotekę „socketlib” będącą interfejsem między gniazdami BSD, a gniazdami Miniksa. Gniazda Miniksa są bardzo proste i nie posiadają wielu funkcjonalności oferowanych przez gniazda BSD, dlatego też są one przez „socketlib” pomijane. Biblioteka implementuje większość z funkcji gniazd BSD za wyjątkiem funkcji select z powodu braku wywołania systemowego select w Miniksie. Biblioteka ta, podobnie jak konwersja OpenSSL, była przetestowana na Miniksie 2.0.2 oraz 2.0.4, ale działa także bez żadnych wymaganych poprawek na Miniksie 2.0.3. Zainstalowanie tej biblioteki i podlinkowanie jej przy kompilacji OpenSSH rozwiązało większość błędów kompilacji. Kod OpenSSH nadal nie kompilował się z powodu paru zależności od kodu standardowo znajdującego się na OpenBSD oraz innych uniksopodobnych systemach. W tym celu pobrałem kod źródłowy systemu OpenBSD 2.7 i zacząłem przenosić część z wymaganych przez OpenSSH funkcjonalności. Bibliotekę „socketlib” zacząłem uzupełniać o resztę zależności kodu OpenSSH od OpenBSD związanych z siecią. Zależności niezwiązane z siecią zgromadziłem w oddzielnej bibliotece „bsdports”. W ten sposób udało mi się uzupełnić brakujące zależności oraz doprowadzić do poprawnej kompilacji klienta SSH. Następnie przystąpiłem do przetestowania klienta SSH poprzez połączenie się z odpowiednio skonfigurowanym serwerem na Ubuntu.

OpenSSH w wersji 1.2.3 korzysta ze starego protokołu SSH w wersji 1. Nowsze wersje OpenSSH wprowadzają stopniowo wsparcie dla protokołu w wersji 2. Serwer OpenSSH 7.6 na Ubuntu 18.04.3 LTS standardowo skonfigurowany jest do obsługi protokołu 2 oraz odrzucania prób połączenia protokołem 1. W pliku konfiguracyjnym serwera można ustawić obsługę obu protokołów. Po konfiguracji serwera przystąpiłem do próby połączenia się z tym serwerem z poziomu maszyny Minix.

Przy próbie połączenia się klientem OpenSSH 1.2.3 do serwera w wersji 7.6 klient otrzymuje komunikat „SSH protocol v.1 is no longer supported”. Okazuje się, że nawet po skonfigurowaniu serwera OpenSSH do obsługi protokołu 1 serwer odmawia nawiązania połączenia z klientem. Obsługa starego protokołu została już w całości usunięta z kodu źródłowego serwera poczynając od wersji 7.6 OpenSSH z powodów bezpieczeństwa [11]. Z tego powodu przestałem konwertować wersję 1.2.3 na Miniksa.

## 6.2. OpenSSH 2.1.1

Z powodu braku obsługi protokołu SSH-2 przez OpenSSH 1.2.3 zacząłem poszukiwać odpowiedniej wersji, która ten protokół obsługuje. Zależało mi przede wszystkim na tym, aby wersja ta w pełni działała z najnowszą wersją dostępną na systemie Ubuntu oraz aby była dostatecznie stara, co zwiększy szansę na jej poprawną kompilację i działanie na starym systemie Minix 2.0.3.

Wersja 2.1.1p4 z roku 2000 zawiera wsparcie dla protokołu SSH-2. W celu upewnienia się, że wersja ta pozwoli na zestawienie połączenia w obie strony z wersją 7.6 pobrałem obraz systemu OpenBSD w wersji 2.7 oraz zainstalowałem go na maszynie wirtualnej. Następnie skonfigurowałem sieć do komunikacji z maszyną Ubuntu. Dystrybucja OpenBSD zawiera w sobie preinstalowany pakiet OpenSSH 2.1 [10]. Przy próbie połączenia się klienta w wersji 2.1 z serwerem w wersji 7.6 otrzymuję komunikat „no matching cipher found”. Wersja 2.1 obsługuje protokół SSH-2, ale nie posiada nowych algorytmów szyfrowania połączenia, szyfrowania kluczy oraz protokołów uzgadniania kluczy dostępnych na wersji 7.6. Wersja 7.6 standardowo korzysta tylko z algorytmów wybranych jako domyślne, ale pliki konfiguracyjne pozwalają na rozszerzenie tej listy o starsze algorytmy. Klient OpenSSH pozwala także na przekazywanie tych algorytmów poprzez argumenty. Wiele z starszych algorytmów nie znajduje się na liście domyślnych z powodów bezpieczeństwa, są one już słabe bądź wrażliwe na ataki.

Za pomocą argumentów polecenia SSH wybierając algorytm szyfrowania kluczy DSS, algorytm szyfrowania połączenia 3DES oraz protokół uzgadniania kluczy Diffiego-Hellmana możemy nawiązać połączenie z klienta OpenSSH 7.6 na Ubuntu z serwerem w wersji 2.1 na OpenBSD. Dodając te same algorytmy do listy domyślnych w pliku konfiguracyjnym serwera OpenSSH 7.6 na Ubuntu możemy także nawiązać połączenie w drugą stronę. OpenSSH 2.1 zatem w pełni współgra z wersją 7.6, o ile zdecydujemy się na wybranie odpowiednich algorytmów.

Z punktu widzenia bezpieczeństwa używanie algorytmów standardowo wyłączonych na serwerze może okazać się niebezpieczne. Oczywiście jeżeli celem tej pracy byłoby zapewnienie jak największego bezpieczeństwa należałoby zainteresować się najnowszą wersją OpenSSH. Konwersja wersji 2.1 ma służyć wyłącznie do użytku na poziomie maszyn wirtualnych, których interfejsy sieciowe emulowane przez oprogramowanie do wirtualizacji nie pozwalają na dostęp z komputera. Ruch sieciowy do maszyn wirtualnych z zewnątrz jest blokowany. Konwersja nowszej wersji OpenSSH może być jedną z perspektyw rozwoju pracy inżynierskiej.

Po potwierdzeniu kompatybilności obu wersji przystąpiłem do konwersji. OpenSSH 2.1.1 podobnie jak wersja 1.2.3 posiada dłuższe niż 14 znaków nazwy plików, także i tutaj należało dokonać inteligentnego skrócenia nazw przydługich do unikalnych krótszych.

OpenSSH 2.1.1 do kompilacji wymaga biblioteki Zlib oraz biblioteki OpenSSL w wersji co najmniej 0.9.5a. Obie dostępne na Miniksa konwersje tych bibliotek spełniają minimalne wymagania.

Aby zapewnić przenośność oraz kompilację OpenSSH 2.1.1 na wielu systemach operacyjnych standardowo korzysta z narzędzia GNU Autoconf. Narzędzie to generuje plik *configure*. Plik ten jest skryptem powłoki systemowej, który wykrywa elementy systemu operacyjnego oraz oprogramowanie na systemie w celu odpowiedniego skonfigurowania kompilacji. Program na wyjściu generuje plik *Makefile* używany do skompilowania programu. Narzędzie *configure* sprawdza np. istniejące mechanizmy logowania w systemie, typy danych, funkcje biblioteki standardowej. Elementy, których system operacyjny nie posiada mogą zostać wykryte oraz zaimplementowane na poziomie kodu OpenSSH. Dla przykładu plik nagłówkowy języka C *bsd-snprintf.h* definiuje funkcję *snprintf()*

oraz `vsnprintf()` biblioteki standardowej, jeżeli narzędzie `configure` nie wykryje ich w systemie. Pozwala to na kompilację warunkową elementów brakujących w systemie, na których chcemy skompilować program.

Skrypt `configure` standardowo wykrywa rodzaj systemu oraz jego podstawowe elementy, ale nie posiada on konfiguracji dla systemu Minix. W celu wygenerowania pliku `Makefile` do kompilacji OpenSSH na Miniksie napisałem skrypt `config.minix`, który uruchamia narzędzie `configure` z parametrami, które zapewnią najbliższą kompatybilność z Miniksem:

```
#!/bin/sh
./configure --host=i386-minix \
  • --with-cflags="-wo -D_POSIX_SOURCE -D_MINIX_SOURCE -D_MINIX" \
  • --prefix=/usr/local --without-pam
```

Opcje kompilacji `-D_POSIX_SOURCE`, `-D_MINIX_SOURCE`, `-D_MINIX` włączają rozszerzenia UNIX i Minix. Opcja `--prefix` wskazuje na miejsce instalacji oprogramowania, a opcja `--without-pam` informuje konfigurator o niekorzystaniu z mechanizmu Linux PAM nieistniejącego na Miniksie. Po wykonaniu skryptu `configure` pliki wyjściowe wymagały jeszcze dodatkowych zmian. W pliku `Makefile` należało do bibliotek linkowanych w czasie kompilacji dodać napisane przeze mnie biblioteki `socketlib` oraz `bsdports`. Skrypt `configure` nie był w stanie wykryć wszystkich potrzebnych modułów systemu Minix, dlatego też plik `config.h`, który zawiera definicje posiadanego na systemie oprogramowania także musiał zostać ręcznie przeedytowany. Na początku pliku dodałem kompilację warunkową `#ifdef _MINIX` zawierającą dodatkowe definicje potrzebne do poprawnej kompilacji programu.

W następnym kroku zacząłem dalej rozszerzać funkcjonalności bibliotek `socketlib` oraz `bsdports` o brakujący na Miniksie kod OpenBSD potrzebny do kompilacji. Wiedząc, że OpenBSD 2.7 standardowo zawiera w sobie działające OpenSSH 2.1, to zacząłem opierać wszelki dodatkowy kod na tej wersji OpenBSD. Ważną częścią brakującej w Miniksie 2.0.3 funkcjonalności jest wywołanie systemowe `select`. Pozwala ono na monitorowanie wielu deskryptorów plików w oczekiwaniu na gotowość co najmniej jednego. Jest ono szeroko używane w wielu aplikacjach sieciowych. W repozytorium oprogramowania na Miniksa 2 możemy znaleźć dwie niezależne od siebie implementacje tego wywołania: dla wersji 2.0.2 oraz 2.0.3.

Wywołanie dla Miniksa 2.0.3 jest rozpowszechniane w formie paczki zawierającej łatkę do kodu źródłowego jądra systemu oraz testy. Przystąpiłem do instalacji wywołania. Po zaaplikowaniu łatek, rekompilacji jądra oraz restarcie systemu potwierdziłem poprawne wykonywanie się testów.

Po instalacji wszystkich powyżej wymaganych bibliotek udało się skompilować OpenSSH. Przystąpiłem do testowania klienta i serwera OpenSSH 2.1.1 w komunikacji z OpenSSH 7.6 na Ubuntu. Testy ujawniły duży błąd prowadzący do popłochu jądra systemu, którego źródłem jest implementacja wywołania systemowego `select` dla Miniksa 2.0.3. Z poziomu użytkownika błąd wygląda na losowy, aplikacja powoduje błąd systemu w trudnych do określenia momentach pracy systemu. Następnie spędziłem niemało czasu studiując i analizując tą implementację w celu wykrycia przyczyny błędu. Niestety nie udało mi się zlokalizować źródła problemu. Postanowiłem porzucić pracę nad tą implementacją wywołania `select`, a przetestować implementację przygotowaną na Miniksa 2.0.2.

W celach testowych przygotowałem maszynę wirtualną z zainstalowanym Miniksem 2.0.2. Następnie przystąpiłem do instalacji wszelkiego potrzebnego oprogramowania: sterownika karty sieciowej AMD Lance, bibliotek `Zlib`, `OpenSSL`, `socketlib`, `bsdports` oraz wywołania `select`.

Po przygotowaniu maszyny oraz konfiguracji sieci podjąłem się kompilacji OpenSSH 2.1.1. Programy OpenSSH skompilowały się pomyślnie. Podczas testów komunikacji przekonwertowanej wersji 2.1.1 z wersją 7.6 stwierdzono w większości poprawne działanie aplikacji klienta i serwera. Nie występują błędy powodujące zawieszenie się systemu. Po stwierdzeniu poprawnego działania wywołania systemowego `select` przystąpiłem do jego konwersji na Miniksa 2.0.3.

Refaktoryzacje kodu w jądrze 2.0.3 wymuszają ostrożność przy konwersji wywołania systemowego. Poza niewielkimi zmianami w niskopoziomowych mechanizmach jądra większość kodu wywołania `select` pozostaje bez zmian. Po prącochłonnej konwersji wywołania `select` oraz jego przetestowaniu udało mi się potwierdzić jego poprawne działanie na Miniksie 2.0.3. Korzystając z tej implementacji wywołania `select` klient oraz serwer OpenSSH nie powodują zawieszenia się systemu. Pozostała ilość czasu spędziłem na poprawianiu drobnych błędów w działaniu OpenSSH oraz wymaganych bibliotekach.

Wynikiem pracy przy konwersji OpenSSH 2.1.1 jest działający klient i serwer OpenSSH. Programy mogą porozumiewać się z klientami/serwerami OpenSSH 7.6 działającymi na systemie Ubuntu 18.04.3 LTS. Stwierdzono w większości prawidłowe zachowanie oprogramowania, z wyjątkiem dwóch błędów po stronie serwera OpenSSH 2.1.1. Pierwszym z nich jest wymóg dodatkowego wciśnięcia klawisza „Enter” przy zakończeniu sesji przez klienta łączącego się z serwerem OpenSSH 2.1.1. Drugim błędem jest brak działania programu SCP przy próbie połączenia z serwerem na Miniksie. Polecenie SCP wywołane na Miniksie działa poprawnie i pozwala na kopiowanie plików w obie strony. Z powodu braku czasu nie zdążyłem rozwiązać tych problemów. Priorytetem było dla mnie przygotowanie środowiska do pracy na laboratorium, dlatego też na tym etapie zakończyłem swoją pracę nad konwersją OpenSSH 2.1.1 na Miniksa oraz reszty oprogramowania potrzebnego do jego poprawnego działania i kompilacji. Wyniki wykonanej w tym rozdziale pracy zawierają się w maszynie wirtualnej przygotowanej do środowiska na laboratorium. Konwersja wywołania systemowego `select`, biblioteki `socketlib` oraz `bsdports` zostały zawarte na płycie CD w katalogach `kernel_ports`, `socketlib` oraz `bsdports`. Kod źródłowy konwersji OpenSSH 2.1.1 na Miniksa 2.0.3 wraz z gotową konfiguracją pod ten system został zawarty w katalogu `ssh`. W następnym rozdziale opisuję przygotowane środowisko do pracy na laboratorium.



## 7. Środowisko do pracy na laboratorium

Końcowym wynikiem mojej pracy inżynierskiej jest przygotowane środowisko do pracy nad zadaniami laboratorium przedmiotu Systemy Operacyjne. Główne założone przeze mnie cele to łatwość instalacji, przenośność rozwiązania oraz wspieranie wielu systemów do wirtualizacji.

Środowisko wspiera następujące systemy do wirtualizacji:

- VirtualBox
- VMWare
- Qemu

Przetestowano także i stwierdzono niemożliwość działania środowiska na oprogramowaniu HyperV oraz Bochs. System HyperV nie posiada wsparcia dla sterownika karty sieciowej zainstalowanej na Miniksie, co eliminuje wsparcie sieci takiej maszyny, a oprogramowanie Bochs nie było w stanie poprawnie uruchomić maszyny wirtualnej z systemem Minix.

Dla każdego z systemów wirtualizacji przygotowano po dwie skonfigurowane maszyny wirtualne. Wraz z nimi dołączone zostały instrukcje instalacji dla każdego systemu oraz rozwiązania częstych problemów.

Proces instalacji środowiska dla wszystkich trzech systemów przebiega podobnie. Użytkownik kopiuje katalogi maszyn wirtualnych dla wybranego przez siebie oprogramowania do wirtualizacji. Następnie kopiuje dysk wirtualny zawierający system Minix 2.0.3 oraz pobiera dysk instalacyjny systemu Ubuntu (dysk wirtualny systemu Ubuntu z powodu swojej objętości oraz potrzeby przygotowania go dla każdego systemu do wirtualizacji z osobna z powodu drobnych niekompatybilności nie został przygotowany). W następnym kroku użytkownik instaluje system Ubuntu oraz przeprowadza jego podstawową konfigurację. Po instalacji można uruchomić obie maszyny i zacząć korzystać ze środowiska.

Obie maszyny wirtualne standardowo skonfigurowane są do istnienia w jednej sieci w celu możliwości nawiązania połączenia między sobą. Interfejsy kart sieciowych obu maszyn przydzielone mają stałe adresy IP oraz aliasy dla tych adresów o nazwach „minix” i „ubuntu”. Dzięki takiemu rozwiązaniu użytkownik może w prosty sposób używać narzędzi do komunikacji sieciowej nie przejmując się zmiennymi adresami.

Przygotowany dysk wirtualny maszyny Minix zawiera:

- Skonfigurowany do obsługi sieci system operacyjny Minix 2.0.3 z zaaplikowanymi konwersjami sterownika karty sieciowej AMD Lance oraz wywołania systemowego select
- Biblioteki OpenSSL i Zlib
- Oprogramowanie OpenSSH 2.1.1p4
- Socketlib – biblioteka implementująca gniazda sieciowe BSD
- Bsdports – biblioteka zawierająca konwersje kodu źródłowego systemu OpenBSD 2.7

Rozwiązanie powinno być wygodne w użyciu poprzez zapewnienie komunikacji sieciowej między maszynami. Środowisko jest także bardzo przenośne, użytkownik może skopiować katalog maszyny wirtualnej Miniksa na której pracował na laboratorium i kontynuować pracę we własnym zakresie na własnym sprzęcie. Wsparcie trzech popularnych systemów do wirtualizacji VirtualBox, VMWare oraz Qemu zapewnia dużą kompatybilność, oraz umożliwia użytkownikowi na pracę w znanym już sobie systemie.

## 7.1. Zawartość płyty CD

Na płycie CD znajdują się następujące katalogi:

- `virtual_machines` – zawiera przygotowane maszyny wirtualne oraz dysk wirtualny z systemem Minix 2.0.3
- `sources` – zawiera kod źródłowy rozwiązań opracowanych w poprzednich rozdziałach: łąkę na jądro systemu zawierającą konwersję sterownika AMD Lance oraz wywołania systemowego `select`, biblioteki `socketlib`, `bsdports` oraz konwersję oprogramowania OpenSSH 2.1.1p4 na Miniksa 2.0.3
- `minix_install` – zawiera pliki przydatne przy ręcznej instalacji Miniksa 2.0.3, w ich skład wchodzi: oryginalna paczka instalacyjna Miniksa 2.0.3, spreparowany obraz dyskietki instalacyjnej, obraz dysku wirtualnego z resztą plików instalacyjnych oraz obraz dysku wirtualnego ze świeżo zainstalowanym systemem Minix 2.0.3

Katalog `virtual_machines` zawiera podkatalogi dla każdego ze środowisk do wirtualizacji zawierające przygotowane maszyny wirtualne wraz z instrukcjami instalacji oraz katalog `disk_images` zawierający obraz dysku wirtualnego systemu Minix 2.0.3.

Katalog `minix_install` zawiera pliki przydatne przy ręcznej instalacji Miniksa 2.0.3, w ich skład wchodzi: oryginalna paczka instalacyjna Miniksa 2.0.3, spreparowany obraz dyskietki instalacyjnej, obraz dysku wirtualnego z resztą plików instalacyjnych oraz obraz dysku wirtualnego z zainstalowanym systemem Minix 2.0.3.

Katalog `sources` zawiera kod źródłowy rozwiązań opracowanych w poprzednich rozdziałach: łąkę na jądro systemu zawierającą konwersję sterownika AMD Lance oraz wywołania systemowego `select`, biblioteki `socketlib`, `bsdports` oraz konwersję oprogramowania OpenSSH 2.1.1p4 na Miniksa 2.0.3.



## 8. Instalacja środowiska do pracy na laboratorium

W rozdziale tym opisuję przebieg instalacji przygotowanego w ramach tej pracy środowiska. Streszczone instrukcje instalacji znajdują się także w plikach tekstowych w katalogach na płycie CD.

W pierwszej kolejności użytkownik musi zdecydować się, z którego systemu do wirtualizacji chce skorzystać, wyboru dokonuje spośród trzech: VMWare, VirtualBox, Qemu. Wybór systemu do wirtualizacji nie powinien mieć większego wpływu na sposób użycia środowiska po instalacji. Zaleca się korzystanie ze znanego sobie systemu do wirtualizacji. Jeśli użytkownik nie miał doświadczenia z tymi programami wcześniej, zaleca się skorzystanie z rozwiązania na VMWare. Przed instalacją środowiska należy jeszcze wyposażyć się w obraz płyty instalacyjnej systemu Ubuntu Desktop dostępnej na oficjalnej stronie dystrybucji. Zaleca się pobranie Ubuntu w wersji 18.04 LTS.

### 8.1. Instalacja pod VMWare

Należy zainstalować oprogramowanie VMWare Player na komputerze docelowym, o ile nie jest już ono zainstalowane. Instalacja różni się między systemami, proszę zapoznać się ze sposobem instalacji programu VMWare Player na swoim systemie.

Należy przygotować sobie katalog, w którym przechowywane będą maszyny wirtualne programu VMWare Player. Na systemach z rodziny Linux dobrą lokalizacją jest stworzenie katalogu o nazwie *vmware* w katalogu */home/user* danego użytkownika, gdzie *user* to nazwa użytkownika. Następnie należy skopiować katalogi zawierające maszyny wirtualne *virtual\_machines/vmware/minix* oraz *virtual\_machines/vmware/ubuntu* z płyty CD do katalogu docelowego (w przykładzie katalogiem tym jest */home/user/vmware*). Następnie kopiujemy dysk wirtualny z systemem Minix *virtual\_machines/disk\_images/minix.vmdk* z płyty CD do katalogu z maszyną wirtualną Minix (W przykładzie */home/user/vmware/minix*).

Otwieramy program VMWare Player. Dodajemy obie maszyny wirtualne do biblioteki za pomocą opcji *File > Open a Virtual Machine*. Wchodzimy w ustawienia maszyny Ubuntu i do urządzeń maszyny tworzymy nowy dysk wirtualny oraz dodajemy napęd płyt CD/DVD, montujemy w nim obraz płyty instalacyjnej Ubuntu. Następnie uruchamiamy maszynę Ubuntu i instalujemy system. Konfigurujemy system według rozdziału 10.4. Po instalacji Ubuntu maszyny powinny być gotowe do działania, w celu rozpoczęcia pracy można uruchomić obie maszyny.

Przy pierwszym uruchomieniu każdej z maszyn VMWare może pokazać nam komunikat z informacją o przeniesieniu bądź skopiowaniu maszyny, należy wybrać wtedy opcje „I Moved it”.

### 8.2. Instalacja pod VirtualBoxem

Należy zainstalować oprogramowanie VirtualBox na komputerze docelowym, o ile nie jest już ono zainstalowane. Instalacja różni się między systemami, proszę zapoznać się ze sposobem instalacji programu VirtualBox na swoim systemie.

Należy przygotować sobie katalog, w którym przechowywane będą maszyny wirtualne programu VirtualBox. Na systemach z rodziny Linux dobrą lokalizacją jest stworzenie katalogu o nazwie *virtualbox* w katalogu */home/user* danego użytkownika, gdzie *user* to nazwa użytkownika. Następnie należy skopiować katalogi zawierające maszyny wirtualne *virtual\_machines/virtualbox/minix* oraz *virtual\_machines/virtualbox/ubuntu* z płyty CD do katalogu docelowego (w przykładzie katalogiem tym jest */home/user/virtualbox*). Następnie kopiujemy dysk wirtualny z systemem Minix

*virtual\_machines/disk\_images/minix.vmdk* z płyty CD do katalogu z maszyną wirtualną Minix (W przykładzie */home/user/virtualbox/minix*). Przed przystąpieniem do uruchomienia maszyn należy uruchomić skrypt *create\_network.sh* z katalogu *virtual\_machines/virtualbox/create\_network.sh* płyty CD. Skrypt ten stworzy sieć NAT w środowisku maszyn VirtualBox, z której korzystać będą nasze maszyny wirtualne.

Otwieramy program VirtualBox. Dodajemy obie maszyny wirtualne do biblioteki za pomocą przycisku *Add*. Wchodzimy w ustawienia maszyny Ubuntu i do urządzeń maszyny pod zakładką *Storage*, dodajemy urządzenie *Controller: SATA*, wraz z nim tworzymy nowy dysk wirtualny, następnie dodajemy urządzenie *Controler: IDE*, pod którym dodajemy napęd płyt CD/DVD wybierając opcję *Add Optical Drive*, montujemy w nim obraz płyty instalacyjnej Ubuntu. Następnie uruchamiamy maszynę Ubuntu i instalujemy system. Konfigurujemy system według rozdziału 10.4. Po instalacji Ubuntu maszyny powinny być gotowe do działania, w celu rozpoczęcia pracy można uruchomić obie maszyny.

### 8.3. Instalacja pod Qemu

Należy zainstalować oprogramowanie Qemu na komputerze docelowym, o ile nie jest już ono zainstalowane. Instalacja różni się między systemami, proszę zapoznać się ze sposobem instalacji programu Qemu na swoim systemie.

Należy kopiować katalog */virtual\_machines/Qemu* na dysk swojego komputera do wygodnego miejsca, np. do katalogu domowego użytkownika. Następnie kopiujemy dysk wirtualny z systemem Minix *virtual\_machines/disk\_images/minix.vmdk* z płyty CD do katalogu, który w poprzednim kroku skopiowaliśmy (w przykładzie */home/user/qemu*). Płytę instalacyjną systemu Ubuntu przenosimy do skopiowanego folderu i zmieniamy nazwę tego pliku na *ubuntu\_install.iso*.

Uruchamiamy skrypt *install\_ubuntu.sh*. Uruchomi on maszynę wirtualną i zamontuje płytę instalacyjną Ubuntu. Instalujemy system Ubuntu, konfiguracja opisana jest w rozdziale 10.4. Po instalacji Ubuntu maszynę możemy zamknąć. Środowisko powinno być gotowe. Do uruchamiania obu maszyn korzystamy ze skryptów *start\_ubuntu.sh* oraz *start\_minix.sh*.

### 8.4. Konfiguracja systemu Ubuntu

Po instalacji systemu Ubuntu należy przeprowadzić jeszcze jego krótką konfigurację. W pierwszej kolejności należy upewnić się, że interfejs sieciowy maszyny jest dobrze skonfigurowany: dla VMWare opcja *NAT*, dla VirtualBoxa opcja *Nat network > natnet1*. Po uruchomieniu systemu należy ustawić stały adres IP maszyny w sieci o wartości 192.168.43.17. Należy zapoznać się z dokumentacją posiadanej wersji Ubuntu w celu dowiedzenia się jak to zrobić.

Na koniec pliku `/etc/ssh/ssh_config` dodać następujące linie:

```
Host 192.168.43.16
    KexAlgorithms Diffie-hellman-group1-sha1
    HostKeyAlgorithms ssh-dss
    Ciphers 3des-cbc
Host minix
    KexAlgorithms Diffie-hellman-group1-sha1
    HostKeyAlgorithms ssh-dss
    Ciphers 3des-cbc
```

Pozwoli to na korzystanie ze starszych algorytmów SSH, ale tylko do komunikacji z maszyną Minix. Ostatnim krokiem jest dodanie aliasu dla maszyny Minix w pliku `/etc/hosts`:

```
192.168.43.16 minix
```

Dzięki temu będziemy mogli w wygodny sposób porozumiewać się z maszyną Minix. Przy obu uruchomionych maszynach polecenie `ssh root@minix` powinno pozwolić nam zalogować się jako użytkownik `root` na maszynie Minix.



## 9. Podsumowanie

Omówione w mojej pracy zagadnienia związane z systemem operacyjnym Minix oraz środowiskiem wspierającym pracę na laboratorium przedmiotu Systemy Operacyjne są bardzo szerokie. Podczas mojej pracy skupiłem się na dostarczeniu działającego i łatwego w instalacji środowiska do pracy z systemem Minix, zapewniającego komunikację sieciową będącą kluczem do uproszczenia i zwiększenia wygody pracy na laboratorium. Wyniki pracy są bazą, która może być dalej rozwijana przez chętne osoby.

Pierwszą z rzeczy wartą rozwoju może być dalsze rozszerzanie dostępnego oprogramowania na system Minix 2.0.3, np. o wygodniejsze edytory tekstu takie jak Vim. Dzięki zainstalowanym bibliotekom socketlib, BSDports oraz implementacji wywołania systemowego select konwertowanie aplikacji sieciowych może być o wiele prostsze. Sama konwersja oprogramowania OpenSSH 2.1.1p4 nadal zawiera błędy, które także można próbować naprawić.

Poza rozwojem oprogramowania maszyny Miniksa można zastanowić się także nad zautomatyzowaniem procesu instalacji maszyn wirtualnych poprzez napisanie skryptów instalacyjnych wykonujących operacje za użytkownika. Kolejną możliwością jest ubogacenie maszyn wirtualnych o skrypty, które mogłyby masowo kopiować pliki źródłowe jądra Miniksa potrzebne na danym ćwiczeniu laboratoryjnym w obie strony między maszyną Minix, a maszyną Ubuntu.

Cele pracy inżynierskiej zostały osiągnięte. Wytworzone środowisko powinno zapewnić wyższy komfort pracy na laboratorium przedmiotu Systemy Operacyjne. Dostarczone maszyny wirtualne, wraz z komunikacją sieciową oraz podstawowymi narzędziami do jej obsługi eliminują wiele z problemów, z którymi studenci musieli mierzyć się dotychczas na laboratorium oraz zapewniają większą wydajność pracy nad zadaniami.



## 10. Bibliografia

- [1] Changes from Minix 2 to Minix 3.  
<https://wiki.minix3.org/doku.php?id=www:documentation:improvements> [dostęp 30 stycznia 2020]
- [2] Compiling and using Minix network support.  
<https://minix1.woodhull.com/faq/netinst.html> [dostęp 30 stycznia 2020]
- [3] Minix 2. <https://minix1.woodhull.com/> [dostęp 30 stycznia 2020]
- [4] Minix 2.0.3 changes. <https://www.minix-vmd.org/pub/minix/2.0.3/changes.txt> [dostęp 30 stycznia 2020]
- [5] Minix 2.0.4 AMD Lance and Intel Pro driver.  
<https://minix1.woodhull.com/pub/contrib/204ether.txt> [dostęp 30 stycznia 2020]
- [6] Minix 2.0.4 changes.  
<https://minix1.woodhull.com/current/2.0.4/changes.txt> [dostęp 30 stycznia 2020]
- [7] Minix 3.  
<https://wiki.minix3.org/doku.php?id=www:documentation:read-more> [dostęp 30 stycznia 2020]
- [8] Minix 3 Releases.  
<https://wiki.minix3.org/doku.php?id=releases:previous> [dostęp 30 stycznia 2020]
- [9] OpenBSD 2.7. <https://www.openbsd.org/27.html> [dostęp 30 stycznia 2020]
- [10] OpenSSH. <https://www.openssh.com/features.html> [dostęp 30 stycznia 2020]
- [11] OpenSSH 7.6. <https://www.openssh.com/txt/release-7.6> [dostęp 30 stycznia 2020]
- [12] OpenSSH Portable. <https://www.openssh.com/portable.html> [dostęp 30 stycznia 2020]





## 11. Spis rysunków

Rysunek 1: Vmware Player – tworzenie nowej maszyny wirtualnej.....	10
Rysunek 2: VMWare Player – pytanie o medium instalacyjne systemu.....	10
Rysunek 3: VMWare Player – wybór typu systemu operacyjnego .....	11
Rysunek 4: VMWare Player – wybór nazwy maszyny wirtualnej .....	11
Rysunek 5: VMWare Player – konfiguracja dysku wirtualnego maszyny.....	12
Rysunek 6: VMWare Player – podsumowanie.....	12
Rysunek 7: VMWare Player – ekran konfiguracji sprzętu maszyny wirtualnej.....	13
Rysunek 8: VMWare Player – dodawanie czytnika dyskietek do maszyny .....	13
Rysunek 9: VMWare – wybór obrazu dyskiетки instalacyjnej .....	14
Rysunek 10: Minix – monitor inicjacji systemu Minix.....	14
Rysunek 11: Minix – wybór urządzenia z plikami katalogu /usr .....	15
Rysunek 12: Minix – ekran logowania .....	15
Rysunek 13: Minix – instrukcje do instalacji systemu Minix .....	16
Rysunek 14: Minix – instrukcja partycjonowania dysku .....	16
Rysunek 15: Minix – interfejs programu part.....	17
Rysunek 16: Minix – wypełnienie tabeli partycji dysku.....	17
Rysunek 17: Minix – wybór stworzonej partycji.....	18
Rysunek 18: Minix – podsumowanie podziału partycji.....	18
Rysunek 19: VMWare Player – tworzenie drugiego dysku wirtualnego .....	19
Rysunek 20: Minix – montowanie dysku z plikami instalacyjnymi .....	20
Rysunek 21: Minix – instalacja plików odpowiedzialnych za obsługę sieci.....	21



## **12. Spis tabel**

Tabela 1: Wsparcie dla kart sieciowych w poszczególnych wersjach systemu Minix ..... 6

Tabela 2: Wsparcie emulacji kart sieciowych przez oprogramowanie do wirtualizacji ..... 7