# Kurs administrowania systemem Linux 2024

Lista zadań na pracownię nr 12

Na zajęcia 20 i 29 maja 2024

**Zadanie 1 (wersja *legacy*, 2 pkt).** Zainstaluj emulator systemu DOS, np. popularny DOSBox (`DOSBOX.COM`) lub (nieco zapomniany) DOSEMU (`DOSEMU.ORG`) albo prawdziwy system DOS, np. FreeDOS (`freedos.org`), na emulatorze trybu rzeczywistego procesora x86, np. na Qemu, VirtualBoksie lub PCEmu albo na rzeczywistej maszynie (można zrobić *dual boot* dla Linuksa i FreeDOS, np. za pomocą `memdisk`). Zainstaluj w nim podstawowe narzędzia, w tym „the masterpiece of assembler programming" — The Volkov Commander (`vvv.kiev.ua`). Ze znanych serwisów oferujących klasyczne gry (`DOSGAMES.COM`, `classicdosgames.com`) wybierz kilka, zainstaluj je i uruchom.[1] Przygotuj krótką prezentację możliwości systemu DOS i jego emulatorów. Przejrzyj jakiś dobry podręcznik programowania dla systemu DOS (np. Ray Duncan, *Advanced MS-DOS Programming: The Microsoft Guide for Assembly Language and C Programmers*, Microsoft Press 1988) i przygotuj porównanie systemu DOS i systemów uniksowych, w szczególności opisz zadania systemu DOS i odpowiedz na pytania:

- Czy DOS zarządza pamięcią?
- Czy pamięć jest zwirtualizowana?
- Czy jest w DOS-ie ochrona pamięci?
- Czym jest proces w DOS-ie?
- Czy DOS dokonuje podziału czasu pomiędzy procesami?
- Jak odbywa się zarządzanie procesami?
- Jak jest zorganizowana obsługa wywołań systemowych?
- Jak architektura procesora 8088 wpłynęła na strukturę systemu DOS?

**Zadanie 2 (wersja *współczesna*, 2 pkt).** Skopiuj z GitHuba:

`https://github.com/Cacodemon345/uefidoom`

port popularnej gry „Doom" na „system operacyjny" UEFI. Skompiluj go i dodaj jako opcję do Gruba tak, abyś mógł go uruchamiać *zamiast* Linuksa. Uwaga: port nie jest rozwijany od 2020 roku i ponoć nie jest wolny od błędów! W Githubie jest dostępna skompilowana wersja binarna.

**Zadanie 3 (1 pkt).** Przygotuj obraz dysku `disk.img` rozmiaru 1 TiB (2147483648 sektorów) i podziel go na partycje GPT zgodnie z poniższą tabelą:

| Partycja | Początek | Koniec | Typ | Nazwa |
|---|---|---|---|---|
| 1 | 34 | 2047 | BIOS boot | `GNU Grub 2` |
| 2 | 2048 | 262143 | EFI System Partition | `ESP` |
| 3 | 262144 | 2097151 | Linux filesystem data | `Linux /boot` |
| 4 | 2097152 | 1065353215 | Linux LUKS | `Linux rootfs` |
| 5 | 1065353216 | 1073741823 | Linux swap | `Linux & OpenBSD swap` |
| 6 | 1073741824 | 2147483614 | OpenBSD disklabel | `OpenBSD rootfs` |

[1] Wasz prowadzący z czasów swoich studiów pamięta „Prince of Persia". „Doom" pojawił się później.

Identyfikatory dysku i partycji powinny być następujące:

| Identyfikator dysku: `7C54FB1F-0CEE-497F-9557-BCE60937DEF6` | |
|---|---|
| Partycja | UUID |
| 1 | `CC053A2A-29B4-4D05-9E49-F2D4CD0C3000` |
| 2 | `B7E80F66-60D8-41C8-AD03-F54810C0CE42` |
| 3 | `81633C0B-0C11-4346-A2E5-5E7B46601BCB` |
| 4 | `C3191E5B-D22B-4377-AF3E-C899A17E4CCA` |
| 5 | `B056DC21-CD09-408A-A009-C28DE1E33DEC` |
| 6 | `A5A73AB0-E418-4405-8466-70636A7F35B0` |

Identyfikatory typów partycji są następujące:

| | |
|---|---|
| BIOS boot | `21686148-6449-6E6F-744E-656564454649` |
| EFI System Partition | `C12A7328-F81F-11D2-BA4B-00A0C93EC93B` |
| Linux filesystem data | `0FC63DAF-8483-4772-8E79-3D69D8477DE4` |
| Linux LUKS | `CA7D7CCB-63ED-4C53-861C-1742536059CC` |
| Linux swap | `0657FD6D-A4AB-43C4-84E5-0933C84B4F4F` |
| OpenBSD disklabel | `824CC7A0-36A8-11E3-890A-952519AD3F61` |

Ponieważ OpenBSD uruchamiany za pomocą *legacy BIOS* oczekuje partycji MS-DOS, przygotuj hybrydowy sektor MBR zawierający następujące partycje:

| Identyfikator dysku: `0x7C54FB1F` | | | | | |
|---|---|---|---|---|---|
| Partycja | Początek | Koniec | Typ | CHS początku | CHS końca |
| 1 | 1 | 1073741823 | ee | 0/0/2 | 1023/254/63 |
| 2 | 1073741824 | 2147483614 | a6 | 1023/254/63 | 1023/254/63 |
| 3 | pusta | | | | |
| 4 | 2147483615 | 2147483647 | ee | 1023/254/63 | 1023/254/63 |

**Zadanie 4 (3 pkt).** Załóż na pendrivie tablicę GPT zawierającą następujące partycje:

| Partycja | Początek | Koniec | Typ | Zawartość | Przeznaczenie |
|---|---|---|---|---|---|
| 1 | 34 | 2047 | unused | — | — |
| 2 | 2048 | 131071 | ESP | FAT32 | jądro Gruba (UEFI) |
| 3 | 131072 | do końca dysku | LUKS | ext2 w LUKS1 | rootfs GRUBa |

Za pomocą polecenia `grub-mkfont(1)` utwórz plik `terminus12x24.pf2` zawierający czcionkę Terminus $12 \times 24$. Utwórz archiwum TAR zawierające tę czcionkę w katalogu `fonts/`. Za pomocą polecenia `grub-mkimage(1)` utwórz jądro Grub-a zawierające *memdisk* z czcionką Terminus i następujący plik konfiguracyjny:

```
set root=(memdisk)
set prefix=($root)/
loadfont terminus12x24
set gfxmode=auto
set gfxpayload=keep
set gfxterm_font=terminus12x24
terminal_output gfxterm
set home_uuid=UUID partycji 3
cryptomount -u $home_uuid
set root=cryptouuid/$home_uuid
set prefix=($root)/grub
configfile grub.cfg
```

Wygenerowane jądro gruba umieść w katalogu `\EFI\boot\bootx64.efi` w partycji 2. Będzie to jedyny nie zaszyfrowany plik w Twoim pendrivie. W partycji 3 załóż kontener zaszyfrowany LUKS1, sformatuj w nim system plików `ext2`, zamontuj tymczasowo w katalogu `/mnt` i skopiuj do katalogu `/mnt/grub/x86_64-efi/` biblioteki Gruba, np. tak:

```
rsync -av --exclude=monolithic --exclude=kernel.img --exclude=config.h \
   --exclude=fdt.lst /usr/lib/grub/x86_64-efi/ /mnt/grub/x86_64-efi/
```

Do katalogu `/mnt/boot` skopiuj jądro i initramfs systemu, który chcesz uruchamiać za pomocą tego zaszyfrowanego pendrive'a. Utwórz plik `/mnt/grub/grub.cfg` zawierający deklaracje odpowiedniego menu.

**Zadanie 5 (2 pkt).** Grub2 niezbyt nadaje się do rozszyfrowywania partycji LUKS — kłopotem jest *key stretching* — Grub2 działa sekwencyjnie na jednym wątku procesora i nie umie korzystać z rozkazów AES-NI procesora. Odszyfrowanie jest kilka-kilkanaście razy wolniejsze, niż podczas normalnej pracy polecenia `cryptsetup` w Linuksie. Rozwiązanie z poprzedniego zadania możesz poprawić w następujący sposób. Niech rootfs uruchamianego systemu będzie skonfigurowany jak w zadaniu 3 z listy 10. Partycja 3 pendrive'a jest zaszyfrowana w trybie `plain` (bez nagłówka), a jej klucz szyfrujący jest zapisany jawnie w pierwszych 64 bajtach zaszyfrowanego rootfs-a. W środku zaszyfrowanego kontenera znajduje się system ext4, który nie używa pierwszych 1024 bajtów dysku (jednak wypełnia je zerami podczas formatowania, więc konfigurację pendrive'a należy przeprowadzić *po* sformatowaniu rootfs-a). Polecenie `plainmount` (dostępne w wersji 12.2 Grub-a[2]) pozwala odszyfrować tak skonfigurowaną partycję. Znajdzie on tam, jak w poprzednim zadaniu, swój katalog domowy oraz jądro i initramfs uruchamianego systemu. Jeśli na pendrive'a przeniesiemy też, jak w zadaniu 3 z listy 10, header rootfs-a, to musimy odpowiednio zmienić initramfs (łatwe).

**Zadanie 6 (2 pkt).** W tym zadaniu skonfigurujemy pendrive do *alternatywnego* uruchomienia naszego normalnie skonfigurowanego systemu zainstalowanego na twardym dysku (wyposażonego we własny bootloader itd.). Załóż na pendrivie tablicę GPT zawierającą cztery partycje:

| Partycja | Początek | Koniec | Typ | Zawartość | Przeznaczenie |
|---|---|---|---|---|---|
| 1 | 34 | 2047 | BIOS boot | binarna | jądro Gruba (MBR) |
| 2 | 2048 | 131071 | ESP | FAT32 | jądro Gruba (UEFI) |
| 3 | 131072 | 524287 | Linux filesystem | ext2 | rootfs GRUBa |
| 4 | 524288 | do końca dysku | LUKS | LUKS/ext4 | kontener na dane |

Załóż na partycjach wskazane systemy plików. Za pomocą polecenia `grub-install(8)` zainstaluj jądro Gruba dwukrotnie, w wersji `i386-pc` oraz `x86_64-efi` (opcja `--target`). Katalogiem domowym Gruba uczyń katalog główny systemu na partycji 3 (opcja `--boot-directory`). Pamiętaj o wyłączeniu aktualizacji konfiguracji płyty głównej (opcja `--no-nvram`) i instalacji w wersji przenośnej (opcja `--removable`). W katalogu `/grub/` na partycji 3 przygotuj plik `grub.conf` zawierający konfigurację wspólną dla obu wersji Gruba. Grub powinien się uruchomić zarówno wówczas, gdy jest uruchamiany przez *legacy BIOS* jak i przez UEFI.[3] Uruchom Gruba z pendrive'a na swoim komputerze. Uruchom powłokę Gruba i wpisz odpowiednie polecenia `linux` i `initrd` tak, by po wydaniu polecenia `boot` na komputerze został uruchomiony zainstalowany na nim system. Dodaj do pliku `grub.conf` na pendrivie następujące opcje:

1. uruchomienie systemu z twardego dysku,
2. *chainload bootloadera* z twardego dysku,
3. *chainload* jądra z twardego dysku (zadziała tylko w wersji EFI),

[2]We wcześniejszych wersjach Grub-a należało łatać kod źródłowy Grub-a. Gotowe łaty są dostępne na stronie `grub.johnlane.ie`

[3]Niektóre dystrybucje, np. FreeBSD, robią tak zawsze: instalują oba bootloadery, mając nadzieję, że przynajmniej jeden z nich zadziała. Z mojego (skromnego) doświadczenia wynika jednak, że jest niewiele komputerów dostatecznie starych, by nie miały wsparcia dla UEFI i jednocześnie dostatecznie nowych, by ich BIOS umiał uruchamiać system z pendrive'a.

4. ponownego rozruchu,
5. wyłączenia komputera,
6. wejścia do BIOS-u,
7. inne, np. uruchomienie Doom-a.

Dodaj do katalogu `/etc/kernel/postinst.d/` *hook*, który automatycznie kopiuje najnowsze pliki `vmlinuz` i `initrd.img` do partycji ESP po każdej aktualizacji jądra tak, by pozycja 3 powyższego menu zawsze uruchamiała najnowsze jądro. Zapoznaj się następnie z dystrybucją `Super Grub2` i przygotuj krótkie omówienie jej możliwości.

**Zadanie 7 (2 pkt).** Zapoznaj się z dokumentami:

`https://www.gnu.org/software/grub/manual/grub/grub.html`
`http://wiki.rosalab.ru/en/index.php/Grub2_theme_tutorial`
`http://wiki.rosalab.ru/en/index.php/Grub2_theme_/_reference`

i przygotuj krótkie omówienie sposobu konfigurowania interfejsu graficznego Gruba. Przygotuj następnie estetyczną szatę graficzną dla Gruba. Punktem wyjścia mogą być domyślne szaty Starfield lub Breeze. Wiele inspiracji można znaleźć na `deviantart.com`. Twoja szata nie musi być zupełnie oryginalna, ale nie możesz się ograniczyć jedynie do skopiowania gotowego rozwiązania.

**Zadanie 8 (3 pkt).** Zapoznaj się z artykułem: Greig Paul, James Irvine, *Take Control of Your PC with UEFI Secure Boot*, Linux Journal (257):58–72 Sep. 2015. Przeczytaj też wiki Arch-a: *Unified Extensible Firmware Interface* i *Secure Boot* (wiki Gentoo też może być pomocne). Przygotuj krótkie omówienie sposobu generowania i instalowania *Platform Key* na komputerze oraz podpisywania instalowanego oprogramowania. Zapoznaj się i przygotuj krótkie omówienie pakietu *Open Virtual Machine Firmware* i sposobu jego integracji z Qemu. Postępuj zgodnie z opisem z artykułu Paula i Irvine'a: wygeneruj własny PK, usuń stary PK z platformy i zainstaluj własny. Podpisz Gruba własnym kluczem i przetestuj w Qemu, że platforma UEFI właściwie realizuje Secure Boot.

**Zadanie 9 (3 pkt).** Zapoznaj się i przygotuj krótkie omówienie sposobów weryfikacji sygnatur jądra przez Gruba (dla wersji 2.12 — rozdział 19 podręcznika programu). Za pomocą GPG wygeneruj parę kluczy i podpisz nimi pliki Gruba i uruchamiane jądro. Sprawdź, że Grub faktycznie weryfikuje poprawność sygnatur.

**Zadanie 10 (2 pkt).** Przygotuj omówienie aplikacji `shim` rozwijanej przez *Red Hat Bootloader Team* (zob. `github.com/rhboot`). Zapoznaj się ze sposobem konfiguracji Secure Boot w Debianie Stable i skonfiguruj instalację Debiana tak, aby uruchamiała się poprawnie z włączonym Secure Boot (zobacz `wiki.debian.org/SecureBoot`). Przygotuj krótkie omówienie.

**Zadanie 11 (3 pkt).** Zapoznaj się i przygotuj krótkie omówienie pakietu GNU EFI. Napisz prostą aplikację EFI o nazwie `myboot.efi`, która po uruchomieniu wyświetla menu:

1. Boot (przekazuje sterowanie do prawdziwego bootloadera),
2. License (wypisuje na ekranie plik zawierający treść licencji GPL 3),
3. Reboot (powoduje ponowne uruchomienie płyty głównej),
4. Halt (zatrzymuje system).

Ścieżki do prawdziwego bootloadera i pliku tekstowego mogą być „zaszyte" w aplikacji. Warto przeczytać: `www.rodsbooks.com/efi-programming`.

**Zadanie 12 (1 pkt).** Zainstaluj w swojej partycji ESP UEFI Shell v. 2. Skonfiguruj płytę główną żeby go uruchamiała, jeśli żadna inna aplikacja EFI nie jest dostępna. Dodaj go też do menu swojego Grub-a. Przeczytaj jego dokumentację i przetestuj jego możliwości, zarówno pracy interaktywnej, jak i możliwości skryptowania.