



AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Implementacja podstaw systemu operacyjnego zgodnego z POSIX

Cechy systemu

- Jądro: monolityczne z dynamicznie ładowanymi modułami
- Zgodność ze standardem multiboot
- Funkcje systemowe: w większości zgodne z standardem POSIX
- Biblioteka standardowa: port biblioteki newlib
- Aplikacje: pliki wykonywalne w formacie ELF
- Struktura katalogów: zgodna z FHS (Filesystem Hierarchy Standard)

Cechy systemu cd.

Obsługiwane procesory:

- i686 oraz lepsze (wersja 32 bitowa)
- amd64 (wersja 64 bitowa)

Obsługiwane urządzenia:

- Dyski: ATA/IDE
- Napędy: ATAPI
- Karta graficzne: VGA (tryb tekstowy)
- Klawiatury: AT
- Szyna PCI

Obsługiwane systemy plików:

- Ext2 (w trybie tylko do odczytu)
- Ramfs (system plików przechowujący informacje w pamięci RAM)

Jądro systemowe

- Jądro wielozadaniowe, wieloużytkownikowe
- Obsługa pamięci wirtualnej
- Przełączanie zadań poprzez zamianę stosów
- Jądro jest wywłaszczalne
- Moduły w formacie relokowalnych plików ELF
- System plików oparty o architekturę węzłów
- Podział urządzeń na znakowe i blokowe

Biblioteki

- Na chwilę obecną aplikacje linkowane statycznie
- Implementacja standardowej biblioteki języka C: newlib
- Przeniesione popularne biblioteki jak readline czy ncurses

Aplikacje i procesy

- Pliki wykonywalne w formacie ELF
- Każdy proces działający w swojej przestrzeni adresowej
- Synchronizacja przy pomocy blokad wirujących, semaforów i muteksów
- Stworzone odpowiedniki popularnych aplikacji jak cat czy ls
- Przeniesione popularne aplikacje z systemów UNIXowych jak bash, binutils, gcc, less, nano i inne

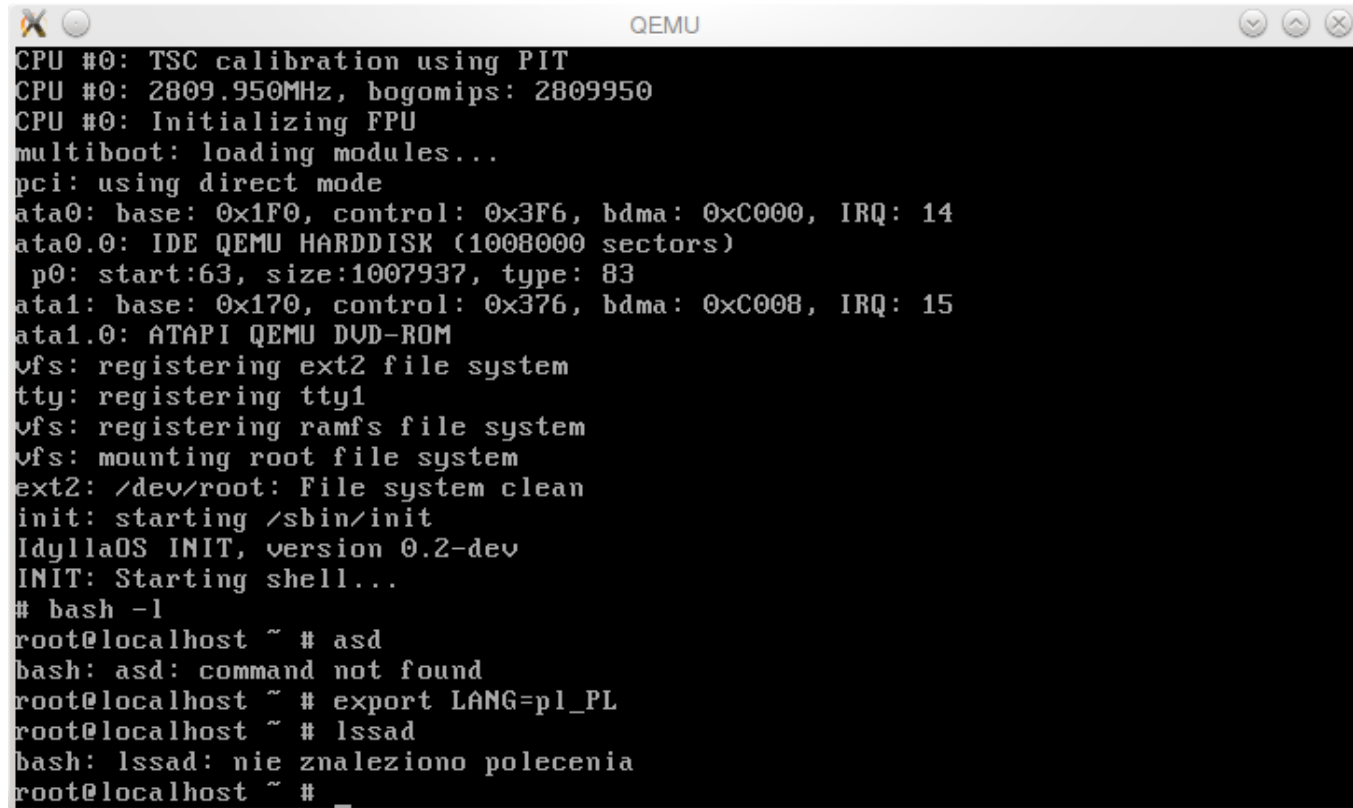
Proces ładowania systemu

- Program ładujący (bootloader) ładuje jądro oraz moduły do pamięci
- Po załadowaniu program skacze do funkcji `_start` jądra
- Następuje inicjalizacja podstawowych urządzeń (procesor, pamięć, etc.) oraz wszystkie moduły zależne od architektury
- Inicjowane są pozostałe moduły (jak manager pamięci wirtualnej)
- Moduły zostają zlinkowane z jądrem

Proces ładowania systemu cd.

- Jądro tworzy nowy proces INIT a samo przechodzi w proces bezczynności
- Proces INIT montuje główny system plików
- Następnie wykonywany jest program /sbin/init
- Na chwilę obecną program /sbin/init uruchamia powłokę (/bin/bash)
- Po załadowaniu powłoki można rozpocząć pracę

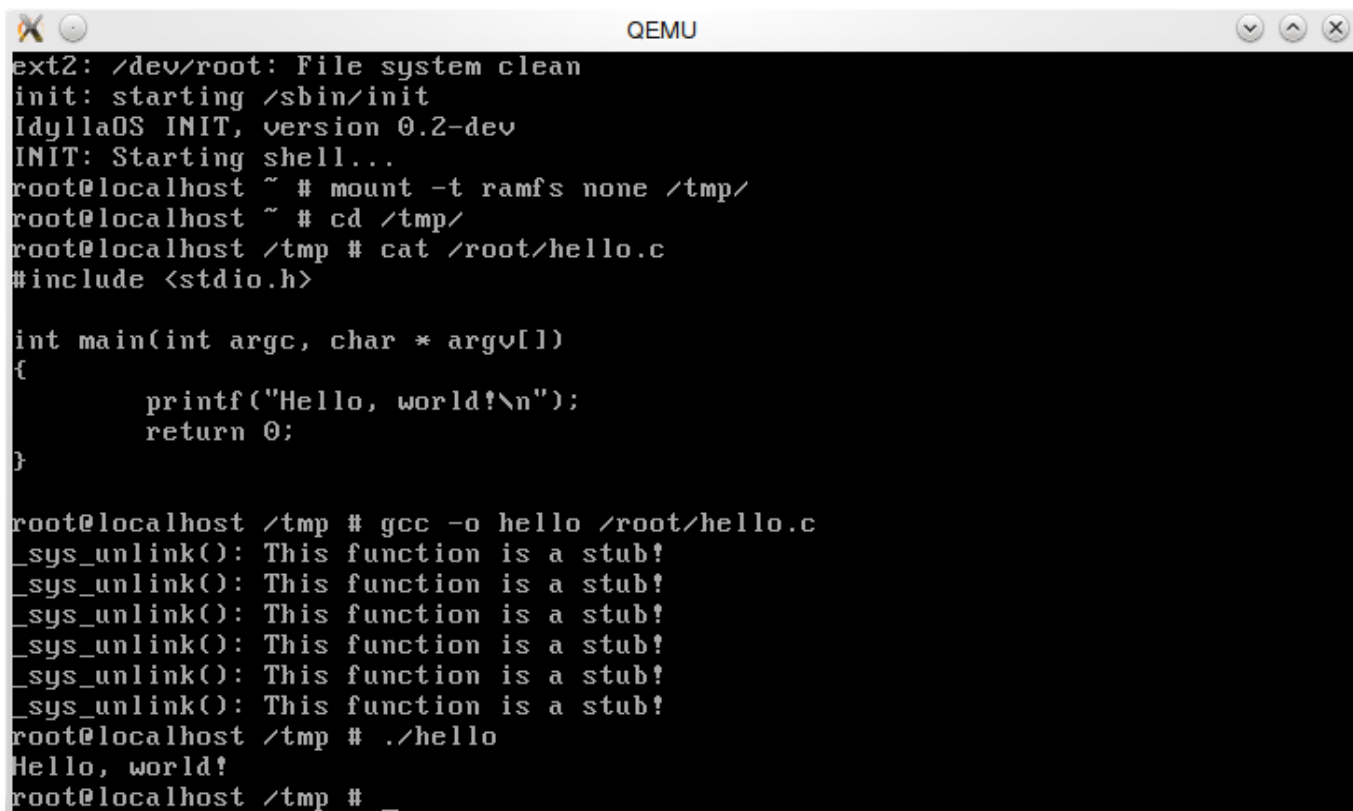
Praca w systemie



```
QEMU
CPU #0: TSC calibration using PIT
CPU #0: 2809.950MHz, bogomips: 2809950
CPU #0: Initializing FPU
multiboot: loading modules...
pci: using direct mode
ata0: base: 0x1F0, control: 0x3F6, bdma: 0xC000, IRQ: 14
ata0.0: IDE QEMU HARDDISK (1008000 sectors)
  p0: start:63, size:1007937, type: 83
ata1: base: 0x170, control: 0x376, bdma: 0xC008, IRQ: 15
ata1.0: ATAPI QEMU DVD-ROM
vfs: registering ext2 file system
tty: registering tty1
vfs: registering ramfs file system
vfs: mounting root file system
ext2: /dev/root: File system clean
init: starting /sbin/init
IdyllaOS INIT, version 0.2-dev
INIT: Starting shell...
# bash -l
root@localhost ~ # asd
bash: asd: command not found
root@localhost ~ # export LANG=pl_PL
root@localhost ~ # lssad
bash: lssad: nie znaleziono polecenia
root@localhost ~ #
```

Działająca powłoka bash (możliwość polonizacji systemu poprzez locales)

Praca w systemie



```
ext2: /dev/root: File system clean
init: starting /sbin/init
IdyllaOS INIT, version 0.2-dev
INIT: Starting shell...
root@localhost ~ # mount -t ramfs none /tmp/
root@localhost ~ # cd /tmp/
root@localhost /tmp # cat /root/hello.c
#include <stdio.h>

int main(int argc, char * argv[])
{
    printf("Hello, world!\n");
    return 0;
}

root@localhost /tmp # gcc -o hello /root/hello.c
_sys_unlink(): This function is a stub!
_sys_unlink(): This function is a stub!
_sys_unlink(): This function is a stub!
_sys_unlink(): This function is a stub!
_sys_unlink(): This function is a stub!
_sys_unlink(): This function is a stub!
root@localhost /tmp # ./hello
Hello, world!
root@localhost /tmp #
```

Działający kompilator GCC

Praca w systemie



```
bash-4.1# ls
drwxr-xr-x 2      520 .
drwxr-xr-x 9     1024 ..
bash-4.1# cat /root/test.S
.data
hello:
    .string "Hello world\n"

.globl _start
_start:
    movl    $4,%eax
    movl    $1,%ebx
    movl    $hello,%ecx
    movl    $12,%edx
    int     $0x80

    .halt:
        jmp .halt

bash-4.1# as -o test.o /root/test.S
bash-4.1# ld -o test test.o
_sys_chmod(): This function is a stub!
bash-4.1# ./test
Hello world
```

Działające aplikacje z pakietu GNU binutils

Plany na przyszłość

- Poprawa błędów
- Optymalizacja niektórych fragmentów kodu
- Implementacja nowych funkcji jądra
- Dodanie sterowników do popularnych urządzeń oraz systemów plików
- Implementacja stosu TCP/IP oraz obsługi sieci