

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Semestrální práce

KIV/OS Simulace operačního systému

Plzeň 2020

David Friesecký
A20N0010K
frieseck@students.zcu.cz

Obsah

1	Zadání	2
2	Analýza	4
2.1	Kernel	4
2.1.1	Procesy	5
2.1.2	Vlákna	5
2.2	I/O	6
2.2.1	Souborový systém	6
2.2.2	Shell	6
2.2.3	Roury	7
2.2.4	Přesměrování	7
3	Implementace	8
3.1	Procesy a vlákna	8
3.2	I/O	10
3.2.1	Souborový systém FAT12	10
3.2.2	I/O funkce	10
3.2.3	Roury	10
3.2.4	Přesměrování	10
3.2.5	Shell	11
4	Závěr	13

1 Zadání

- Vytvořte virtuální stroj, který bude simulovat OS
- Součástí bude shell s gramatikou cmd, tj. včetně exit
- Vytvoříte ekvivalenty standardních příkazů a programů
 - echo, cd, dir, md, rd, type, find /v "/c , sort, tasklist, shutdown
 - * cd a dir musí umět relativní cesty
 - * dir musí umět /S
 - * echo musí umět @echo on a off
 - * type musí umět vypsat jak stdin, tak musí umět vypsat soubor
 - Dále vytvoříte programy rgen a freq
 - rgen bude vypisovat náhodně vygenerovaná čísla v plovoucí čárce na stdout, dokud mu nepřijde EOF, ETX, nebo EOT
 - freq bude číst z stdin a sestaví frekvenční tabulku bytů, kterou pak vypíše pro všechny byty s frekvencí větší než 0 ve formátu: "0x
- Implementujte roury a přesměrování
- Nebudete přistupovat na souborový systém, ale použijete simulovaný disk se souborovým systémem FAT12
 - Součástí zadání je obraz diskety FreeDOS 1.2, vůči kterému budou prováděny testy při hodnocení semestrální práce
 - V DOSBoxu ho lze připojit příkazem: `imgmount d fdos1_2_floppy.img -size 512,63,16,1`

Při zpracování tohoto zadání použijte a dále pracujte s kostrou tohoto řešení, kterou najdete v archívu `os_simulator.zip`. Součástí archívu, ve složce `compiled`, je soubory `checker.exe` a `test.exe`. Soubor `checker.exe` je validátor semestrálních prací. Soubor `test.exe` generuje možný testovací vstup pro vaši semestrální práci.

Vaše vypracování si před odevzdáním zkontrolujte programem `checker.exe`. V souboru `checker.ini` si upravte položku `Setup_Environment_Command`,

v sekci General, tak, aby obsahovala cestu dle vaší instalace Visual Studio. Např. vzorové odevzdání otestujete příkazem "compiledbackslashcharvzoro-rove odevzdani.zip", spuštěného v kořenovém adresáři rozbaleného archívu. Odevzdávaný archív nemá obsahovat žádné soubory navíc a program musí úspěšně proběhnout.

2 Analýza

2.1 Kernel

Jádro operačního systému, je program, který je při startu zaveden do operační paměti a zůstává v ní až do jeho ukončení. Po zavedení je jádru předáno řízení, dokončí inicializaci hardwaru. Zajišťuje správu prostředků (procesor, paměť, periferní zařízení).

Jádro může být:

- monolitické – obsahuje veškeré funkce
- mikrojádro – obsahuje jen ty nejdůležitější a zbylé běží v uživatelském prostoru
- hybridní – kombinace obou předchozích

2.1.1 Procesy

Proces je instance běžícího programu a informace o procesu jsou drženy v PCB (Process Control Block). Jádru vlastní tabulku procesů, kde jeden záznam tabulky je jeden PCB, tedy jeden proces.

PCB obsahuje:

- identifikátor procesu – PID
- obsah registrů – např. čítač instrukcí PC (Program Counter – adresa následující strojové instrukce)
- adresní prostor procesu
- priorita
- I/O informace – alokovaná I/O zařízení, seznam otevřených souborů apod.

Proces může nabývat celkem pěti základních stavů „Nový / Vytvořený“, „Připravený“ (čeká na spuštění), „Běžící“, „Blokovaný“ (zablokovan čekáním na I/O akci nebo uvolnění prostředku) a „Ukončený“. Dalšími stavy mohou být „Připravený + Pozastavený“ a „Blokovaný + Pozastavený“.



Obrázek 2.1: Stavy procesů

2.1.2 Vlákna

Ve většině operačních systémů je vlákno součástí konkrétního procesu. Stejně jako proces, i vlákna mají uložené informace v tabulce jako záznamy nazý-

vající se TCB (Thread Controll Block).

TCB obsahuje:

- identifikátor vlákna – TID
- ukazatel na záznam procesu PCB
- ukazatel na zásobník SP (Stack Pointer)
- obsah registrů – např. čítač instrukcí PC (Program Counter – adresa následující strojové instrukce)
- stav vlákna
- I/O informace – alokovaná I/O zařízení, seznam otevřených souborů apod.

2.2 I/O

Jedná se rozhraní vstupní a výstupních funkcí, které slouží ke komunikaci s hardwarem, jako jsou paměťová média (disky), klávesnice, myš apod. Patří sem také komunikace se souborovým systémem.

2.2.1 Souborový systém

Umožňuje správu dat uložených na paměťových médiích. Skládá se z implementace souborového systému (více disků více souborových systému), dále z virtuálního souborového systému (rozhraní nezávislé na konkrétním souborovém systému) a nakonec rozhraní určené ke komunikaci mezi jádrem a virtuálním souborovým systémem.

2.2.2 Shell

Shell je program, který umožňuje uživateli zadávat příkazy. Stará se o kontrolu vstupu a provádí příkazy případně systémovým voláním posílá příkaz jádru.

Systémové volání se používá jako obrana proti neoprávněnému přístupu např. pouze jádro má přístup k souborovému systému. Výsledek systémového volání posílá jádro zpět procesu, který ho volal.

2.2.3 Roury

Roura představuje meziprocesovou komunikaci dvou současně spuštěných procesů, kde výstup jednoho je poslán na vstup druhého. Lze jí reprezentovat souborem kam mají přístup oba procesy a jeden do souboru zapisuje a druhý z něj čte (Producent – Konzument).

2.2.4 Přesměrování

Jedná se o způsob přesměrování vstupu na jiný než standardní výstup např. výpis řetězce do konzole můžeme přesměrovat do souboru. Stejně tak to funguje i opačně např. obsah souboru se použije jako vstup.

3 Implementace

3.1 Procesy a vlákna

Záznam o procesu (PCB) je reprezentován třídou `Process` a jsou ukládány do mapy (tabulky) ve třídě `Process_Controller`. Proměnně třídy `Process`:

- `process_name` – název procesu
- `pid` – identifikátor procesu
- `process_tid` – identifikátor hlavního vlákna
- `state` – stav procesu
- `threads` – mapa resp. tabulka vláken
- `working_dir` – pracovní adresář
- `working_dir_sector` – sektor, na kterém začínají data pracovního adresáře
- `handle_in` – vstup procesu
- `handle_out` – výstup procesu

Záznam o vlákne (TCB) je reprezentován třídou **Thread** a jsou uloženy v tabulce daného procesu, kterému náleží. Proměnné třídy **Thread**:

- **tid** – identifikátor vlákna
- **pid** – identifikátor procesu, kterému vlákno náleží
- **state** – stav vlákna
- **std_tid** – identifikátor vlákna standardní knihovny C++
- **std_thread** – instance vlákna standardní knihovny C++
- **entry_point** – vstupní bod vlákna
- **regs** – registry
- **args** – argumenty
- **exit_code** – návratová hodnota při ukončení procesu
- **cond** – podmínková proměnná
- **mutex** – zámek
- **waked_by_handler** – identifikátor budícího procesu
- **waiting_handles** – mapa vláken (identifikátorů), na které dané vlákno čeká
- **slepted_handles** – mapa uspaných vláken (identifikátorů)
- **terminate_handles** – mapa vláken (identifikátorů) pro ukončení

Nový proces se po spuštění hlavního vlákna přesune do stavu **Ready** a po jeho ukončení do stavu **Exited**.

Nové vlákno se při vytvoření nastaví na **Ready** a po zadání funkce **Start()** se stav nastaví na **Running**. Dalšími funkcemi pro úpravu stavu (činnosti) vlákna jsou **Stop()**, která vláknu nastaví stav **Blocked**, funkce **Join()** vlákno nastaví jako **Exited** a poslední **Restart(waiting_tid)** probudí čekající vlákno.

Třída **Process_Controller** obsahuje funkce pro klonování procesů či vláken a funkce pro jejich synchronizaci **Notify**, **Notify_All** a **Wait_For()**.

3.2 I/O

3.2.1 Souborový systém FAT12

Jako souborový systém byla použita FAT12 (viz Zadání). Hlavní vstupní třídou do souborového systému je `FAT12`, který obsahuje proměnné `boot_sector` (první sektor disku s parametry souborového systému), `drive_id` (identifikátor disku), `table_phys` (8bitová fyzická tabulka na disku) a `table_logic` (12bitová logická tabulka sektorů transformovaná z fyzické).

3.2.2 I/O funkce

V knihovně RTL je několik funkcí určených pro práci se vstupem a výstupem. Funkce z RTL knihovny využívají systémová volání.

- `Open_File()` – otevření souboru / vytvoření souboru
- `Write_File()` – zápis dat na výstup
- `Read_File()` – čtení dat ze vstupu
- `Seek()` – nastavení pozice, které se má číst / zapisovat
- `Close_Handle()` – zavření handlu (zavření souboru)
- `Delete_File()` – smazání souboru

3.2.3 Roury

Pro vytvoření roury je v knihovně RTL implementována funkce `Create_Pipe()`. Vyžaduje dva handly (dva procesy), kde jeden je na vstupu do roury a druhý je na jejím výstupu. Funkcionalita je implementována jako problém Producenta – Konzumenta.

3.2.4 Přesměrování

Součástí implementace je přesměrování vstupu do souboru i ze souboru na výstup. V případě přesměrování do souboru, se při jeho neexistenci soubor vytvoří a zapisuje se do nového. Pokud je nastaveno přesměrování ven ze souboru, tak se kontroluje, zda soubor existuje (musí existovat).

3.2.5 Shell

Parsování a spouštění příkazů

Před voláním příkazů je potřeba vstup zkontrolovat. Vstup rozdělen podle mezer na tokeny a postupně je plněna třída `Command`. První token na vstupu je vždy volaný příkaz a následně se načítají parametry. V případě, že je nalezen operátor roury nebo přesměrování je načítán nový příkaz jako bychom měli nový vstup. Pokud je nalezen operátor přesměrování, tak je zbytek vstupu načten jako jméno souboru.

Po dokončení je příkaz (příkazy) spuštěn a čeká se na jeho dokončení.

cd

Slouží pro změnu pracovního adresáře. Nejprve se zavolá funkce knihovny `RTL_Set_Working_Directory()` s novým adresářem. Pokud je vše v pořádku, tak nový pracovní adresář lze získat příkazem `Get_Working_Directory()`, taktéž z knihovny RTL.

dir

Pro vypísání položek ve vybraném adresáři.

echo

Vypíše vše co předáno jako argument. Pokud je jako argument „on“ / „off“, tak se vypisování zapne / vypne. Pokud je příkaz zavolán bez argumentů, tak vypíše aktuální stav. V případě zadání argumentu „/?“, vypíše se pomocný řetězec.

find

Příkaz vrací počet řádek souboru předaného v argumentech. Argumenty musí být ve formátu „/v "/c nazev_souboru“.

freq

Načítá ze vstupu jednotlivé znaky a z nich vytváří frekvenční tabulku. Co použitý znak to počet výskytů v řetězci.

md

Vytvoření adresáře se zadaným jménem.

rd

Maže položku souborového systému. V případě mazání složky je musí být složka prázdná.

rgen

Generuje náhodná čísla pomocí funkce `rand()` dokud se na vstupu neobjeví EOF nebo není stisknut ENTER. Implementace je rozdělena na dvě vlákna, kde jedno generuje čísla a druhé naslouchá na vstupu.

shutdown

Zavolá funkci z knihovny RTL `Shutdown()` a ukončí simulátor operačního systému.

sort

Seřadí a vypíše vstup. Nejdříve rozdělí vstup na řádky a ty následně seřadí pomocí standardní funkce „`sort()`“.

tasklist

Vypíše informace o probíhajících procesech, které jsou ukládány do souboru „`procfs`“. Ten je otevřen, vypsán jeho obsah a zase zavřen.

type

Načítá data ze standardního vstupu nebo ze souboru do string jazyka C++. Na konci celý řetězec vypíše na výstup.

4 Závěr

Pro vytvoření simulátoru bylo vycházeno z dodané kostry. Program při spuštění nevykazuje žádné chyby, ale při zadání příkazu vznikne výjimka při „condition_variable.wait()“. Pravděpodobně i ze stejného důvodu práci nelze zkontrolovat pomocí programu „checker.exe“.