2. Projekt 2010/2011

Podmínky vypracování

- Projekt implementujte v jazyku C.
- Komentujte zdrojové kódy, programujte přehledně. Součástí hodnocení bude i kvalita zdrojového kódu.
- Kontrolujte, zda se všechny procesy ukončují korektně a při ukončování správně rušíte všechny alokované zdroje (např. pro prostředky System V můžete pomocí příkazu `ipcs` zjistit, jaké zdroje jsou v systému alokovány).
- Dodržujte syntax zadaných jmen, formátu souborů a formátu výstupních dat! Čtěte pozorně zadání a poznámky k vypracování u jednotlivých zadání.
- **Dotazy k zadání:** Veškeré nejasnosti a dotazy řešte pouze prostřednictvím diskuzního fóra *Projekty*.

Podmínky překladu

- Pro překlad používejte nástroj *make*. Součástí odevzdání bude soubor `Makefile`.
- Překlad se provede příkazem `make` v adresáři, kde je umístěn soubor Makefile.
- Zdrojové kódy překládejte s přepínači `-std=gnu99 -Wall -Wextra -Werror -pedantic`.

Odevzdání

- Součástí odevzdání budou pouze soubory se zdrojovými kódy (`*.c`, příp. `*.h`) a soubor `Makefile`. Tyto soubory zabalte do archivu s názvem xlogin.zip, kde xlogin je váš login.
- Archiv vytvořte tak, aby po rozbalení byl soubor Makefile umístěn ve stejném adresáři jako je archiv.
- Archiv xlogin.zip odevzdejte prostřednictvím informačního systému, termín *Projekt 2*.
- Pokud nebude dodržena forma odevzdání nebo projekt nepůjde přeložit, bude projekt hodnocen 0 body.

Opravy zadání

- Opravy zadání (chyby, překlepy, vyjasnění) jsou v textu zvýrazněny červeně.
- První oprava byla provedena 4.4.2011.

Zadání projektu

Cíl: Implementujte upravenou variantu synchronizačního problému *spícího holiče*. V holičství je jeden holič a Q židlí v čekárně. Když holič ostříhá zákazníka, zkontroluje čekárnu. Pokud zde čeká zákazník, holič ho usadí do svého křesla a ostříhá. Pokud je čekárna prázdná, holič se posadí do svého křesla a usne. Zákazník, který přijde, usedne do čekárny. Pokud holič spí, vzbudí ho a nechá se ostříhat, jinak čeká, až na něj přijde řada. Pokud je čekárna plná (všechny židle Q jsou obsazeny), zákazník odchází neostříhán.

Každému zákazníkovi a holiči odpovídá jeden proces. Pro synchronizaci procesů použijte semafory

(System V nebo POSIX).

Základní proměnné:

```
Q    : počet židlí
GenC : rozsah pro generování zákazníků [ms]
GenB : rozsah pro generování doby obsluhy [ms]
```

Implementační detaily: Po spuštění vytváří hlavní proces jeden podproces pro holiče. Hlavní proces čeká na ukončení všech svých potomků a poté se ukončí. Každý zákazník bude idenfikován celým číslem, začínajícím od 1. Zákazníci jsou vytvářeni v časových intervalech, které jsou generovány náhodně v rozmezí 0 až GenC v ms. Ostříhání zákazníka simulujte uspáním procesu holiče na dobu, která je generována náhodně v rozmezí 0 až GenB v ms. Proces zákazníká čeká, až ho proces holiče informuje o dokončení stříhání po svém vzbuzení. Pro proces zákazníka nepoužívejte aktivní čekání ani sleep.

Výstupy procesů: Každý zákazník a holič poskytuje informace o právě prováděné akci.
Formát výstupů:

- holič kontroluje čekárnu, zda je další zákazník:

  ```
  A: barber: checks
  ```

- holič je připraven přijmout zákazníka z čekárny:

  ```
  A: barber: ready
  ```

- holič ostříhal zákazníka:

  ```
  A: barber: finished
  ```

- zákazník je vytvořen:

  ```
  A: customer C: created
  ```

- zákazník vstupuje do čekárny:

  ```
  A: customer C: enters
  ```

- zákazník usedá do křesla holiče; tiskne se až poté, co je holič připraven (barber: ready):

  ```
  A: customer C: ready
  ```

- zákazník odchází ostříhán; tiskne se až poté, co holič dokončil stříhání (barber: finished):

  ```
  A: customer C: served
  ```

- zákazník odchází neobsloužen:

  ```
  A: customer C: refused
  ```

Popis proměnných ve výstupu:

- **C**: číslo ~~holiče~~ zákazníka

- **A**: pořadové číslo prováděné akce. Akce se číslují od jedničky. Číslo akce je uloženo ve sdílené paměti (C funkce *shmXXX*), před každou akcí se číslo přečte, použije a zpět zapíše nová hodnota.

Poznámky k vypracování:

- Po překladu vznikne spustitelný soubor se jménem *barbers*, který bude umístěn ve stejném adresáři jako soubor Makefile.
- Při spouštění se zadají celá kladná čísla Q, GenC, GenB a N a řetězec F v tomto pořadí (viz ukázka; význam prvních proměnných viz výše). Číslo N vyjadřuje počet celkově vygerenovaných a obsloužených nebo odmítnutých zákazníků (tj. N procesů zákazníka vznikne a zanikne), po kterých se aplikace ukončí (uvolní se všechny zdroje a ukončí se všechny procesy). Řetězec F reprezentuje název souboru, do kterého se budou ukládat generované informace. Pokud se místo názvu uvede znak -, budou se informace vypisovat na standardní výstup.
- Přístup k souboru (zápis informací) musí být výlučný; pokud zapisuje jeden proces a další chce také zapisovat, musí počkat na uvolnění zdroje. Stejně tak musí být zajištěn výlučný přístup k čítači akcí.
- Implementujte problém tak, abyste předešli uváznutí (deadlock) a vyhladovění (starvation).
- Pokud některý se vstupů nebude odpovídat očekávanému formátu, program vytiskne chybové hlášení a ukončí se.
- Pro generování unikátních klíčů (sdílená paměť apod.) použijte funkci ftok.
- C funkce: fork, wait, shmget, shmat, semget, semctl, sem_open, ...

Ukázka spuštění programu:

```
$ ./barbers 10 500 1000 20 barbers.out
```

10 židlí v čekárně, zákazníci se generují náhodně v intervalu 0 až 500 ms, doba obsluhy zákazníka je 0 až 1000 ms a počet celkově vygenerovaných a obsloužených zákazníků je 20. Informace se budou ukládat do souboru barbres.out.

```
$ ./barbers 20 200 800 20 -
```

20 židlí v čekárně, zákazníci se generují náhodně v intervalu 0 až 200 ms, doba obsluhy zákazníka je 0 až 800 ms a počet celkově vygenerovaných a obsloužených zákazníků je 20. Informace se budou vypisovat na standardní výstup.

Ukázka výstupu programu:

```
$ ./barbers 10 0 0 1 -
```

```
1: barber: checks
2: customer 1: created
3: customer 1: enters
4: barber: ready
```

```
5: customer 1: ready
6: barber: finished
7: customer 1: served
```

Pořadí se může lišit v závislosti na časování, ve výše ovedeném příkladu může být např. prohozen první a druhý řádek (samozřejmě se správným číslováním akcí), pokud proces holiče bude plánován později než proces zákazníka. V rámci jednoho procesu musí být zachováno správné pořadí, např. pro zákazníka jsou možné dvě korektní sekvence:

```
created
enters
ready
served
```

nebo

```
created
enters
refused
```

Také musí být zachováno korektní pořadí akcí mezi procesy, např.

- barber ready bude vždy před customer ready,
- barber finished bude vždy před customer ~~finished~~ served,
- mezi barber ready a barber finished může být právě jeden customer ready,
- ...
