2. Úloha IOS (2018/19)

A. Popis úlohy

Implementujte v jazyce C modifikovaný synchronizační problém *River Crossing Problem* (viz kniha The Little Book of Semaphores).

Na řece je jedna vyhlídková loď. Loď může vézt právě jednu skupinu osob, skupinu tvoří právě čtyři osoby. Nelze tedy vézt více ani méně osob než právě čtyři. Osoby jsou rozděleny do dvou kategorií - první kategorie jsou osoby, které pracují v prostředí operačního systému Linux (*hackers*) a druhou kategorii tvoří osoby, které pracují v prostředí operačního systému Windows (*serfs*). Platí, že nelze vézt skupinu osob, kde je pouze jedna osoba ze stejné kategorie (tři hackers a jeden serf nebo jeden hacker a tři serfs); v takovém případě se tato osoba necítí bezpečně. Povolené jsou tedy následující tři kombinace: 4 hackers; 4 serfs; 2 hackers a 2 serfs.

Osoby, které se chtějí přepravit, se shromažďují na molu. Molo má omezenou kapacitu, pokud se nově příchozí osoba již na molo nevejde, odchází pryč a zkusí přijít na molo později. Tento postup se v případě potřeby opakuje, tj. každá osoba nakonec odpluje. Jakmile je možné na molu vytvořit vhodnou skupinu, nastoupí tato skupina na loď a odpluje. Jedna z osob, která nastoupila na loď, se stává kapitánem a řídí loď. Po určité době se loď vrátí zpět, skupina osob vystoupí a odejde pryč. Po dobu plavby (včetně vylodění skupiny) nevstupuje žádná osoba z mola na loď, osoby však mohou přicházet na molo. Pořadí nalodění osob není definováno. Kapitán opouští loď jako poslední, pořadí vylodění ostatních osob není definováno.

B. Podrobná specifikace úlohy

Spuštění

```
$ ./proj2 P H S R W C
```

kde

- `P` je počet osob generovaných v každé kategorii; bude vytvořeno `P` *hackers* a `P` *serfs*.
  `P >= 2 && (P % 2) == 0`
- `H` je maximální hodnota doby (v milisekundách), po které je generován nový proces *hackers*.
  `H >= 0 && H <= 2000`.
- `S` je maximální hodnota doby (v milisekundách), po které je generován nový proces *serfs*.
  `S >= 0 && S <= 2000`.
- `R` je maximální hodnota doby (v milisekundách) plavby.
  `R >= 0 && R <= 2000`.
- `W` je maximální hodnota doby (v milisekundách), po které se osoba vrací zpět na molo (pokud bylo před tím molo plné).
  `W >= 20 && W <= 2000`.
- `C` je kapacita mola. `C >= 5`.
- Všechny parametry jsou celá čísla.

Implementační detaily

- *Pracujte s procesy, ne s vlákny.*
- Hlavní proces vytváří ihned po spuštění dva pomocné procesy pro generování procesů osob stejné kategorie. Poté čeká na ukončení všech procesů, které aplikace vytváří. Jakmile jsou tyto procesy ukončeny, ukončí se i hlavní proces s kódem (exit code) 0.
- Generování procesů
  - *hacker*: pomocný proces generuje procesy pro hackers; každý nový proces je generován po uplynutí náhodné doby z intervalu `<0, H>`; celkem vygeneruje `P` procesů. Pokud platí `H==0`, všechny příslušné procesy se vygenerují ihned.
  - *serf*: pomocný proces generuje procesy pro serfs; každý nový proces je generován po uplynutí náhodné doby z intervalu `<0, S>`; celkem vygeneruje `P` procesů. Pokud platí `S==0`, všechny příslušné procesy se vygenerují ihned.
  - Každý proces bude interně identifikován celým číslem `I`, začínajícím od `1`. Číselná řada je pro každou kategorii procesů zvlášť.
  - Postupně tedy vznikne hlavní proces, dva pomocné procesy a `2*P` procesů osob.
- Každý proces vykonává své akce a současně zapisuje informace o akcích do souboru s názvem `proj2.out`. Součástí výstupních informací o akci je pořadové číslo `A` prováděné akce (viz popis výstupů). Akce se číslují od jedničky.
- Použijte sdílenou paměť pro implementaci čítače akcí a sdílených proměnných nutných pro synchronizaci.
- Použijte semafory pro synchronizaci procesů.
- *Nepoužívejte aktivní čekání (včetně cyklického časového uspání procesu) pro účely synchronizace.*

Chybové stavy

- Pokud některý ze vstupů nebude odpovídat očekávanému formátu nebo bude mimo povolený rozsah, program vytiskne chybové hlášení na standardní chybový výstup, uvolní všechny dosud alokované zdroje a ukončí se s kódem (exit code) `1`.

Popis procesů a jejich výstupů

Poznámka k výstupům

- `A` je pořadové číslo prováděné akce,
- `NAME` je zkratka kategorie příslušného procesu, tj. `HACK` nebo `SERF`,
- `I` je interní identifikátor procesu v rámci příslušné kategorie,
- `NH` je aktuální počet hackers na molu a
- `NS` je aktuální počet serfs na molu.
- Při vyhodnocování výstupu budou ignorovány mezery a tabelátory.

Proces osoby (*hacker* i *serf*)

1. Po spuštění tiskne `A: NAME I: starts`
2. Pokouší se dostat na molo
   (a) Pokud je molo plné (kapacita je naplněna), uspí se proces na náhodnou dobu z intervalu `<20, W>`; před uspáním tiskne `A: NAME I: leaves queue: NH: NS`. Po vzbuzení tiskne `A: NAME I: is back` a pokusí se opět vstoupit na molo (opakuje bod 2).
   (b) Pokud může vstoupit na molo, tiskne `A: NAME I: waits: NH: NS`. *Poznámka k této variantě: Hodnoty* `NH` *a* `NS` *zahrnují i tento proces, který právě vstoupil na molo.*
3. Jakmile je možné vytvořit vhodnou skupinu:
   (a) Čtyři procesy vstoupí na loď, jeden z nich se stává kapitánem.
   (b) Kapitán při vstupu tiskne `A: NAME I: boards: NH: NS`; ostatní procesy netisknou nic.
   (c) Nástup osob proběhne z pohledu ostatních procesů atomicky, tj. počet osob na molu se sníží o 4 v jednom okamžiku; tento stav je garantován po tisku kapitána.
   (d) *Poznámka k této variantě: Hodnoty* `NH` *a* `NS` *reflektují stav po nalodění všech osob.*
4. Plavba je simulována uspáním procesu kapitána na náhodnou dobu z intervalu `<0, R>`. Ostatní členové posádky (members) čekají na vzbuzení procesu kapitána.
5. Po ukončení plavby a vylodění tiskne každý člen posádky (kromě kapitána)
   `A: NAME I: member exits: NH: NS`.
   Kapitán opouští loď jako poslední a tiskne
   `A: NAME I: captain exits: NH: NS`.
6. Proces odchází pryč a ukončí se.
7. Další osoby mohou vstoupit na loď (bod 3) až poté, co poslední osoba ze čtveřice (kapitán) opustí loď a vytiskne příslušnou informaci (bod 5).

C. Podmínky vypracování

Obecné informace

- Projekt implementujte v jazyce C. Komentujte zdrojové kódy, programujte přehledně. Součástí hodnocení bude i kvalita zdrojového kódu.
- Kontrolujte, zda se všechny procesy ukončují korektně a zda při ukončování správně uvolňujete všechny alokované zdroje .
- Dodržujte syntax zadaných jmen, formát souborů a formát výstupních dat. Použijte základní skript pro ověření korektnosti výstupního formátu (dostupný z webu se zadáním). Informace o skriptu jsou uvedeny v komentáři skriptu.
- Dotazy k zadání: Veškeré nejasnosti a dotazy řešte pouze prostřednictvím diskuzního fóra k projektu 2.

Překlad

- Pro překlad používejte nástroj `make`. Součástí odevzdání bude soubor `Makefile`.
- Překlad se provede příkazem `make` v adresáři, kde je umístěn soubor `Makefile`.
- Po překladu vznikne spustitelný soubor se jménem `proj2`, který bude umístěn ve stejném adresáři jako soubor `Makefile`
- Zdrojové kódy překládejte s přepínači `-std=gnu99 -Wall -Wextra -Werror -pedantic`
- Pokud to vaše řešení vyžaduje, lze přidat další přepínače pro linker (např. kvůli semaforům).

Odevzdání

- Součástí odevzdání budou pouze soubory se zdrojovými kódy (`*.c`, `*.h`) a soubor `Makefile`. Tyto soubory zabalte pomocí nástoje zip do archivu s názvem `proj2.zip`.
- Archiv vytvořte tak, aby po rozbalení byl soubor `Makefile` umístěn ve stejném adresáři, jako je archiv.
- Archiv `proj2.zip` odevzdejte prostřednictvím informačního systému, termín *Projekt 2*.
- Pokud nebude dodržena forma odevzdání nebo projekt nepůjde přeložit, bude projekt hodnocen 0 body.
- Archiv odevzdejte pomocí informačního systému v dostatečném předstihu (odevzdaný soubor můžete před vypršením termínu snadno nahradit jeho novější verzí, kdykoliv budete potřebovat).

D. Ukázka výstupů

Ukázka 1

Spuštění

```
$ ./proj2 2 2 2 200 200 5
```

Výstup

```
1         : HACK 1         : starts
2         : HACK 1         : waits                : 1         : 0
3         : SERF 1         : starts
4         : SERF 1         : waits                : 1         : 1
5         : HACK 2         : starts
6         : SERF 2         : starts
7         : HACK 2         : waits                : 2         : 1
8         : SERF 2         : waits                : 2         : 2
9         : SERF 2         : boards               : 0         : 0
10         : HACK 2         : member exits        : 0         : 0
11         : SERF 1         : member exits        : 0         : 0
12         : HACK 1         : member exits        : 0         : 0
13         : SERF 2         : captain exits       : 0         : 0
```

Ukázka 2

Spuštění

```
$ ./proj2 6 0 0 200 200 5
```

Výstup

```
1        : HACK 1        : starts
2        : HACK 1        : waits               : 1        : 0
3        : SERF 1        : starts
4        : SERF 1        : waits               : 1        : 1
5        : HACK 2        : starts
6        : HACK 2        : waits               : 2        : 1
7        : SERF 2        : starts
8        : SERF 2        : waits               : 2        : 2
9        : SERF 5        : starts
10        : HACK 6        : starts
11        : HACK 3        : starts
12        : SERF 6        : starts
13        : SERF 3        : starts
14        : SERF 2        : boards               : 0        : 0
15        : SERF 5        : waits                : 0        : 1
16        : HACK 6        : waits                : 1        : 1
17        : HACK 3        : waits                : 2        : 1
18        : SERF 6        : waits                : 2        : 2
19        : SERF 3        : waits                : 2        : 3
20        : HACK 4        : starts
21        : HACK 4        : leaves queue         : 2        : 3
22        : SERF 4        : starts
23        : SERF 4        : leaves queue         : 2        : 3
24        : HACK 5        : starts
25        : HACK 5        : leaves queue         : 2        : 3
26        : HACK 2        : member exits         : 2        : 3
27        : SERF 1        : member exits         : 2        : 3
28        : HACK 1        : member exits         : 2        : 3
29        : HACK 4        : is back
30        : HACK 4        : leaves queue         : 2        : 3
31        : SERF 4        : is back
32        : SERF 4        : leaves queue         : 2        : 3
33        : SERF 2        : captain exits        : 2        : 3
34        : SERF 6        : boards               : 0        : 1
35        : HACK 5        : is back
36        : HACK 5        : waits                : 1        : 1
37        : HACK 4        : is back
38        : HACK 4        : waits                : 2        : 1
39        : SERF 4        : is back
40        : SERF 4        : waits                : 2        : 2
41        : HACK 3        : member exits         : 2        : 2
42        : HACK 6        : member exits         : 2        : 2
43        : SERF 5        : member exits         : 2        : 2
44        : SERF 6        : captain exits        : 2        : 2
45        : SERF 4        : boards               : 0        : 0
46        : HACK 5        : member exits         : 0        : 0
47        : HACK 4        : member exits         : 0        : 0
48        : SERF 3        : member exits         : 0        : 0
49        : SERF 4        : captain exits        : 0        : 0
```