Semestrální úloha 36OSY 2005

Téma úlohy: **Průběžný sort**
Vypracoval: **Michal Trs**

3. ročník, obor Výpočetní technika, K336 FEL ČVUT,
Karlovo nám. 13, 121 35 Praha 2

Zadání úlohy

Simulujte průběžný sort. Jsou dány procesy P, S a C. Proces P zasílá náhodná čísla procesu S max. rychlostí **v** čísel za jednotku času. Proces S třídí tyto čísla rychlostí **u** komparací za jednotku času a zasílá nejnižší dostupná čísla procesu C (pokud nejsou k dispozici, je proces C uspán). Odeslaná data jsou v procesu S smazána.

Návrh a popis implementace

Dle zadání jsem úlohu řešil pomocí 3 procesů. Pro předávání dat jsem zvolil roury.
První vznikne proces C po spuštění programu, ten vytvoří rouru s deskriptory `fdSC` a vytvoří potomka S. Proces S vytvoří rouru s deskriptory `fdPS` a vytvoří proces P.
Výměna dat je vždy jednosměrná a proto nemůže dojít k časově závislým chybám. Různé výsledky jsou dány parametry programu `c`,`v`,`u`.
Bylo nutné zvážit jak se bude program ukončovat. Zda stiskem klávesy, nebo po určitém čase, nebo po vygenerování určitého počtu čísel. Zvolil jsem poslední možnost, která mi přišla implementačně nejjednodušší.
Pro vlastní řazení jsem implementoval prioritní frontu pomocí spojového seznamu.
Pro splnění zadání „`v(u)` komparací za jednotku času“ jsem po provedení `v(u)` operací zavolal `sleep(1)` a program tak na 1s uspal. Tím, že čas potřebný na `v(u)` operací je mnohem menší než `sleep(1)` je zaručen konstantní časový úsek.

Udělal jsem 2 verze řešení, které se mírně se liší v procesu S.
Obě varianty načtou z roury PS `u` položek (pokud tam jsou, jinak méně) a všechny je přidají do prioritní fronty a tím je setřídí. Rozdíl je v tom, že varianta **`psort_v1`** vybere z fronty všechny položky a pošle je procesu C. To způsobí, že čísla z procesu P jsou po `u` blocích setříděny. Varianta **`psotr_v2`** vybere z prioritní fronty pouze 1 nejmenší položku a tu pošle procesu C. V 1. průchodu je tedy setříděno `u` položek v 2. průchodu (2 x `u`) – 1 položek atd. Když jsou z procesu P načteny všechny položky, je zbytek obsah fronty poslán procesu C.

Přeložení programu:
```
gcc -o run1 psort_v1.c
gcc -o run1 psort_v1.c
```

Spuštění programu:
```
./run1 -c x1 -v x2 -u x3
./run2 -c x1 -v x2 -u x3
```

Parametry: všechny jsou povinné.
c = počet generovaných čísel

v = počet generovaných čísel za jednotku času (1s)
u = počet komparací za jednotku času (1s)

Př.:
```
$ ./run2 -c 20 -v 5 -u 5
1.prvek - hodn: 13
2.prvek - hodn: 10
3.prvek - hodn: 12
4.prvek - hodn: 15
5.prvek - hodn: 19
6.prvek - hodn: 25
7.prvek - hodn: 27
8.prvek - hodn: 37
9.prvek - hodn: 38
10.prvek - hodn: 43
11.prvek - hodn: 49
12.prvek - hodn: 51
13.prvek - hodn: 58
14.prvek - hodn: 60
15.prvek - hodn: 66
16.prvek - hodn: 67
17.prvek - hodn: 83
18.prvek - hodn: 84
19.prvek - hodn: 86
20.prvek - hodn: 89
```

Pro vypsání nápovědy slouží `run -h` nebo `run --help`

Aplikace neočekává žádná vstupní data, pouze vypisuje data na standardní výstup a po provedení c operací je sama ukončena

Závěr

Podmínka `v(u)` operací za jednotku času je řešena pomocí uspání procesu po `v(u)` průchodech pomocí `sleep` na 1 sekundu. Pro zrychlení by šlo použít např. `usleep(time)`, ale musíme zaručit splnění podmínky: čas generování (řazení) << doba uspání pomocí `sleep (usleep)`.

Pokud by nemuselo být splněno omezení v(u) operací za jednotku času, vyplatilo by se implementovat řazení nějak sofistikovaněji (např.: halda). Prioritní fronta má časovou složitost vkládání lineární.

V následující tabulce jsou shrnuty výsledky pro měření pro **c = 20**. Sloupcům u=1,v=1 odpovídá nesetříděné pole (prvek se řadí sám se sebou). Výsledky ve sloupcích u=20,v=20 odpovídají klasickému sortu (vše co je vygenerováno, je najednou seřazeno a posláno na výstup).

v [op/s]	1		5		5		10		20		20	
u [op/s]	1		5		10		5		10		20	
	psort_v1	psort_v2	psort_v1	psort_v2	psort_v1	psort_v2	psort_v1	psort_v2	psort_v1	psort_v2	psort_v1	psort_v2
	38	38	13	13	10	10	13	13	10	10	10	10
	58	58	15	10	12	12	15	10	12	12	12	12
	13	13	38	12	13	13	38	12	13	13	13	13
	15	15	51	15	15	15	51	15	15	15	15	15
	51	51	58	19	19	19	58	19	19	19	19	19
	27	27	10	25	27	25	10	25	27	25	25	25
	10	10	12	27	38	27	12	27	38	27	27	27
	19	19	19	37	51	37	19	37	51	37	37	37
	12	12	27	38	58	38	27	38	58	38	38	38
	86	86	87	43	86	43	86	43	86	43	43	43
	49	49	25	49	25	49	25	49	25	49	49	49
	67	67	49	51	37	51	49	51	37	51	51	51
	84	84	60	58	43	58	60	58	43	58	58	58
	60	60	67	60	49	60	67	60	49	60	60	60
	25	25	84	66	60	66	84	66	60	66	66	66
	43	43	37	67	66	67	37	67	66	67	67	67
	89	89	43	83	67	83	43	83	67	83	83	83
	83	83	66	84	83	84	66	84	83	84	84	84
	37	37	83	86	84	86	83	86	84	86	86	86
	66	66	89	89	89	89	89	89	89	89	89	89
real time [s]	**19,018**	**19,093**	**3,023**	**4,022**	**3,022**	**4,026**	**3,019**	**4,023**	**1,021**	**2,017**	**0,028**	**1,023**

Literatura

Slide k 3. a 4. cvičení z OSY
Příklady na roury k 4.cvičení
http://www.manualy.sk/unixprg/kap9.htm#SEKCE01220