Projekt nr.2
Systemy operacyjne

Prowadzący: mgr inż. Tomasz Kuczyński
Grupa:

1. Mateusz Gryko
2. Marcin Łogwinowicz
3. Grzegorz Niewierowski
4. Filip Ryciuk

Treść:

Temat 1 – czytelnicy i pisarze

Czytelnicy i pisarze. Z czytelni korzysta na okrągło pewna ilość czytelników i pisarzy, przy czym jednocześnie może w niej znajdować się albo dowolna ilość czytelników, albo jeden pisarz, albo nikt - nigdy inaczej. Problem ten ma trzy rozwiązania - z możliwością zagłodzenia pisarzy, z możliwością zagłodzenia czytelników oraz wykluczające zagłodzenie. Napisać:

a) dwa programy symulujące dwa różne rozwiązania tego problemu, bez korzystania ze zmiennych warunkowych
[17 p], lub

b) dwa programy symulujące dwa różne rozwiązania tego problemu, przy czym jeden z nich musi korzystać ze zmiennych warunkowych (condition variable).
[27 p], lub

c) trzy programy symulujące trzy różne rozwiązania tego problemu, przy czym przynajmniej jeden z nich musi korzystać ze zmiennych warunkowych
[34 p].

Ilość wątków pisarzy R i czytelników W można przekazać jako argumenty linii poleceń. Zarówno czytelnicy jak i pisarze wkrótce po opuszczeniu czytelni próbują znów się do niej dostać. Program powinien wypisywać komunikaty według poniższego przykładu:

ReaderQ: 11 WriterQ: 10 [in: R:0 W:1]
Oznacza to, że w kolejce przed czytelnią czeka 10 pisarzy i 11 czytelników a sama czytelnia zajęta jest przez jednego pisarza. Komunikat należy wypisywać w momencie zmiany którejkolwiek z tych wartości.

Funkcje:
- reader - rutyna wątku czytelnika
- writer - rutyna wątku pisarza
- sig_handler_sigusr1 - handler sygnału SIGUSR1 ma bezpiecznie zakończyć program bez wycieków pamięci
- kill -SIGUSR1 [PID] - bezpiecznie zakończenie działania procesu i wszystkich jego wątków.

Zmienne Globalne:

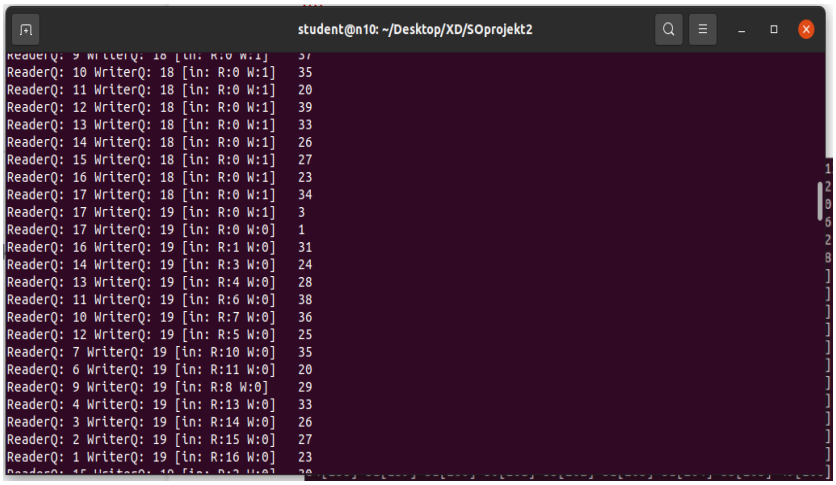
```
pthread_mutex_t mutexR;            //zmienna blokujaca liczbe czytajacych
pthread_mutex_t mutex;            //zmienna blokujaca biblioteke oraz liczbe writerow
pthread_mutex_t mutexQW;           //zmienna blokujaca kolejke writerow
pthread_mutex_t mutexQR;           //zmienna blokujaca kolejke readerow
pthread_mutex_t mutexcheck;        //mutex do sprawdzania wejsc
pthread_cond_t cond;                    //zmienna warunkowa reagujaca na zmiane ilosci writerow
pthread_cond_t condR;                   //zmienna warunkowa reagujaca na zmiane ilosci readerow
int W = 0, R = 0;                  //ilosci writerow i readerow
int *check = NULL;                //tablica do sprawdzania ilosci wejsc osoby
int coutR = 0;                    // ilosc czytelnikow w bibliotece
int coutW = 0;                    // ilosc pisarzy w bibliotece
int end = 0;                     //zmienna bezpiecznego zakonczenia
int queueW=0;                                 //ilosc watkow w kolejce writerow
int queueR=0;                                 //ilosc watkow w kolejce readerow
```

Format wyników: ilość wejść[ID] – id od 0 do W-1 to writerzy, a od W do W+R-1 to readerzy

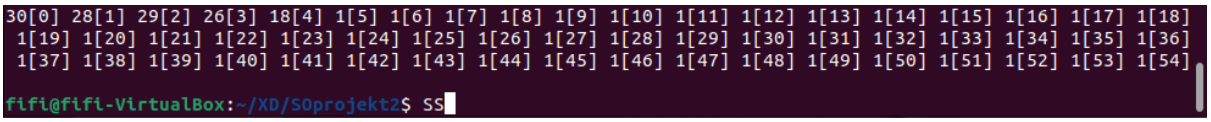
writStarw(50,300):

```
1[0] 2[1] 2[2] 2[3] 2[4] 2[5] 2[6] 2[7] 2[8] 2[9] 2[10] 2[11] 2[12] 2[13] 2[14] 2[15] 2[16] 2[17] 2[18] 2[19] 2[
20] 2[21] 2[22] 2[23] 2[24] 2[25] 2[26] 2[27] 2[28] 2[29] 2[30] 2[31] 2[32] 2[33] 2[34] 2[35] 2[36] 2[37] 2[38]
2[39] 2[40] 2[41] 2[42] 2[43] 2[44] 2[45] 2[46] 2[47] 2[48] 1[49] 51[50] 53[51] 51[52] 50[53] 51[54] 49[55] 52[5
6] 51[57] 47[58] 51[59] 46[60] 49[61] 52[62] 53[63] 52[64] 50[65] 52[66] 50[67] 53[68] 51[69] 54[70] 47[71] 51[7
2] 48[73] 53[74] 53[75] 51[76] 52[77] 50[78] 51[79] 52[80] 51[81] 51[82] 51[83] 51[84] 51[85] 51[86] 52[87] 51[8
8] 54[89] 50[90] 54[91] 51[92] 53[93] 50[94] 52[95] 51[96] 53[97] 51[98] 55[99] 50[100] 52[101] 52[102] 52[103]
55[104] 54[105] 51[106] 49[107] 48[108] 52[109] 53[110] 49[111] 50[112] 49[113] 51[114] 50[115] 53[116] 51[117]
51[118] 47[119] 51[120] 50[121] 51[122] 52[123] 50[124] 52[125] 50[126] 50[127] 52[128] 54[129] 52[130] 50[131]
54[132] 50[133] 52[134] 54[135] 52[136] 49[137] 50[138] 51[139] 50[140] 48[141] 51[142] 52[143] 51[144] 50[145]
52[146] 52[147] 49[148] 52[149] 49[150] 53[151] 53[152] 49[153] 51[154] 51[155] 52[156] 50[157] 51[158] 52[159]
50[160] 51[161] 51[162] 48[163] 48[164] 56[165] 52[166] 51[167] 50[168] 52[169] 51[170] 52[171] 51[172] 49[173]
53[174] 54[175] 50[176] 53[177] 50[178] 52[179] 54[180] 49[181] 52[182] 53[183] 52[184] 49[185] 52[186] 49[187]
49[188] 52[189] 51[190] 51[191] 52[192] 51[193] 51[194] 49[195] 53[196] 52[197] 50[198] 49[199] 49[200] 53[201]
51[202] 50[203] 50[204] 50[205] 52[206] 50[207] 50[208] 48[209] 51[210] 53[211] 52[212] 53[213] 53[214] 48[215]
53[216] 53[217] 51[218] 53[219] 55[220] 57[221] 51[222] 52[223] 51[224] 51[225] 50[226] 52[227] 53[228] 51[229]
52[230] 53[231] 53[232] 55[233] 49[234] 53[235] 50[236] 50[237] 50[238] 50[239] 53[240] 55[241] 50[242] 55[243]
49[244] 53[245] 53[246] 50[247] 49[248] 52[249] 53[250] 55[251] 49[252] 51[253] 53[254] 50[255] 54[256] 52[257]
54[258] 51[259] 51[260] 50[261] 53[262] 51[263] 51[264] 53[265] 49[266] 49[267] 49[268] 50[269] 51[270] 51[271]
48[272] 51[273] 53[274] 50[275] 49[276] 51[277] 52[278] 47[279] 53[280] 51[281] 50[282] 49[283] 51[284] 52[285]
52[286] 50[287] 53[288] 50[289] 50[290] 51[291] 51[292] 52[293] 48[294] 50[295] 48[296] 49[297] 50[298] 49[299]
50[300] 52[301] 49[302] 52[303] 51[304] 50[305] 49[306] 51[307] 50[308] 49[309] 51[310] 53[311] 51[312] 48[313]
51[314] 53[315] 52[316] 53[317] 51[318] 53[319] 49[320] 48[321] 52[322] 52[323] 52[324] 49[325] 51[326] 48[327]
50[328] 50[329] 52[330] 50[331] 52[332] 48[333] 50[334] 50[335] 51[336] 48[337] 53[338] 48[339] 52[340] 54[341]
52[342] 50[343] 55[344] 47[345] 50[346] 52[347] 52[348] 50[349]
fifi@fifi-VirtualBox:~/XD/SOprojekt2$
```

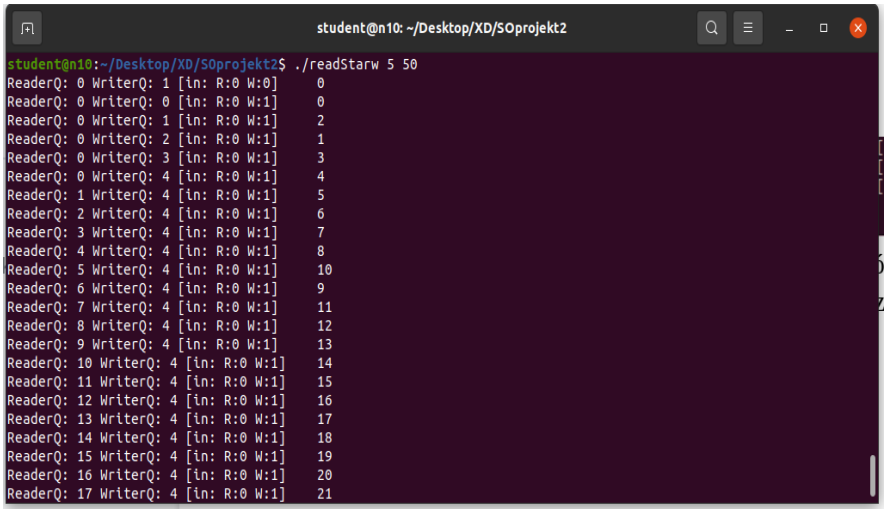
Przy dużej ilości readerów większość writerów weszła 2 razy, przy starcie programu, ponieważ program zaczyna działanie od writerów oraz na koniec programu ponieważ po killu program blokuje wejście do kolejki i czeka na przejście wszystkich elementów przed zakończeniem. Readerzy wchodzili wiele razy.



readStarw(5,50)



W przypadku readStarw od pewnej wartości writerów (~5) readerom uda się wejść tylko raz - przy zakończeniu programu. Writer po wyjściu zdąży wejść z powrotem do kolejki zanim reszcie writerów uda się wejść.

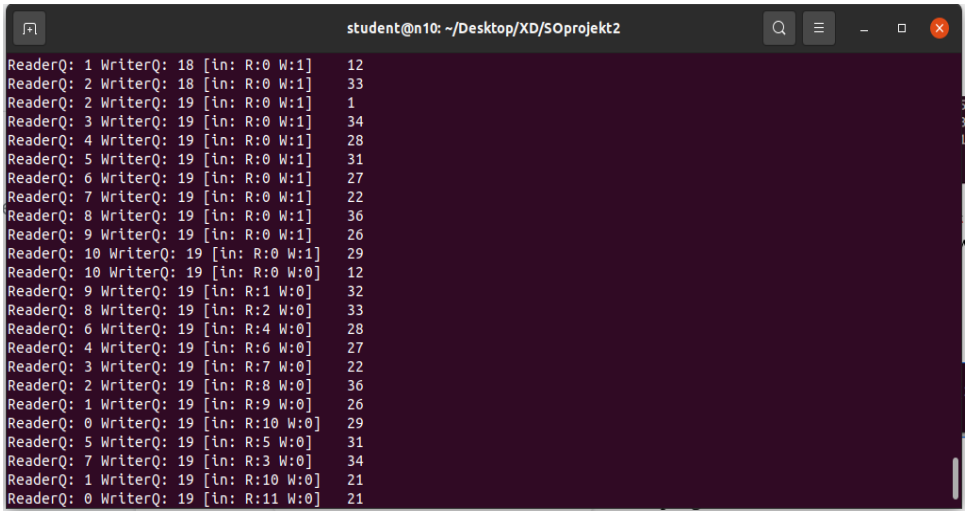


noStarw(20,20)

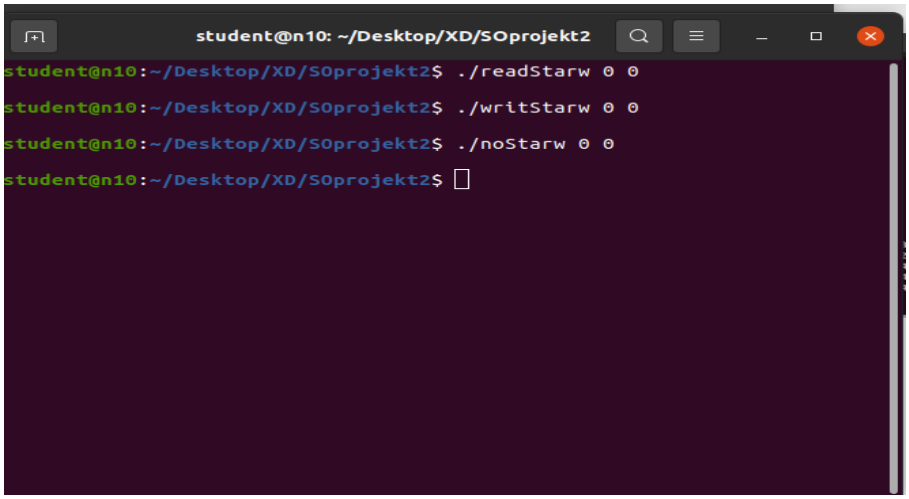
```
8[0] 11[1] 10[2] 11[3] 5[4] 8[5] 4[6] 7[7] 7[8] 9[9] 11[10] 9[11] 10[12] 7[13] 5[14] 13[15] 16[16] 7[17] 9[18]
14[19] 140[20] 136[21] 131[22] 142[23] 137[24] 138[25] 142[26] 136[27] 140[28] 141[29] 140[30] 143[31] 142[32]
141[33] 140[34] 137[35] 138[36] 139[37] 133[38] 133[39]
student@n10:~/Desktop/XD/SOprojekt2$
```

Z upływem czasu zarówno readerzy jak i writerzy wejdą do czytelni. Ani writer ani reader nie zagłodzą się w trakcie działania programu.

Fragment działania programu:



Dla **obu** parametrów równych 0 programy się nie aktywują.



Podział pracy:

Marcin - przypadek readstarw, rutyny reader i writer
Mateusz - przypadek nostarw, rutyny reader i writer
Grzegorz - wszystkie funkcje main
Filip - przypadek writestarw, rutyny reader i writer