

Oskar Sobczyk

Systemy operacyjne – problem palaczy

1. SPIS TREŚCI

1.	Spis treści.....	2
2.	Opis zadania	3
2.1	Problem palaczy tytoniu (ang. The Cigarette-Smokers Problem)	3
2.2	Rozwiązanie problemu	3
3.	Omówienie kodu źródłowego	4
3.1	Proces main.c	4
3.2	Proces valet.c.....	4
3.3	Proces smoker.c.....	5
4.	Testowanie	5

2. OPIS ZADANIA

2.1 PROBLEM PALACZY TYTONIU (ANG. THE CIGARETTE-SMOKERS PROBLEM)

Rozważmy problem, w którym występuje lokaj oraz trzech palaczy. Każdy palacz nieustannie skręca i wypala papierosy. Aby zapalić papierosa, palacz potrzebuje trzech rzeczy: tytoniu, papieru i zapalek. Lokaj posiada nieskończony zapas papieru, tytoniu i zapalek, a palacze: pierwszy ma tytoń, drugi papier, trzeci zapalki. Lokaj losowo wybiera dwa z posiadanych przez siebie zasobów, a palacz posiadający trzeci może wypalić papierosa. Lokaj „nie wie”, który palacz posiada jaki zasób. Agent czeka aż palacz skończy palić i ponownie wystawia dwa różne, losowe składniki. Cykl się powtarza.

2.2 ROZWIĄZANIE PROBLEMU

W celu rozwiązania zadania potrzebne będzie utworzenie czterech procesów (trzech procesów palaczy oraz jednego procesu lokaja). Procesy będą synchronizowane za pomocą czterech semaforów:

sem_valet informuje, czy lokaj może udostępnić składniki,

sem_tobacco informuje, czy palacz z tytoniem może skręcić papierosa,

sem_paper informuje, czy palacz z papierem może skręcić papierosa,

sem_matches informuje, czy palacz z zapalkami może skręcić papierosa.

Proces główny na początku swojego działania utworzy i zamknie semafony. Następnie za pomocą polecenia ***fork()*** utworzy cztery procesy. W trzech z nich zostaną wywołane programy palaczy a w jednym lokaja. Proces lokaja czeka, aż ***sem_valet*** zostanie zwolniony, a następnie losuje liczbę z przedziału $[0,2]$ odpowiadającą kombinacji brakujących składników i zwalnia semafor odpowiedniego palacza. Program palacza czeka, aż odpowiedni semafor zostanie zwolniony i jeśli jest dostępny, to wypisuje komunikat o skręceniu papierosa, a następnie zwalnia ***sem_valet***.

3. OMÓWIENIE KODU ŹRÓDŁOWEGO

3.1 PROCES MAIN.C

W początkowej części (Zrzut ekranu 1) main.c następuje utworzenie oraz zamknięcie semaforów.

```
sem_valet=sem_open("/sem_valet", O_CREAT, (S_IRWXU | S_IRWXG | S_IRWXO), 1);
sem_tobacco=sem_open("/sem_tobacco", O_CREAT, (S_IRWXU | S_IRWXG | S_IRWXO), 0);
sem_paper=sem_open("/sem_paper", O_CREAT, (S_IRWXU | S_IRWXG | S_IRWXO), 0);
sem_matches=sem_open("/sem_matches", O_CREAT, (S_IRWXU | S_IRWXG | S_IRWXO), 0);

sem_close(sem_valet);
sem_close(sem_tobacco);
sem_close(sem_paper);
sem_close(sem_matches);
```

Zrzut ekranu 1

W dalszej części (Zrzut ekranu 2) zajmujemy się klonowaniem procesu za pomocą funkcji **fork()**.

```
valet = fork();
if (valet == 0)
{
    char *args[] = {"valet", 0};
    char *envir[] = {NULL};
    execve("valet", args, envir);
}
else
{
    smoker_tobacco = fork();
    if (smoker_tobacco == 0)
    {
        char *args[] = {"smoker", "1", 0};
        char *envir[] = {NULL};
        execve("smoker", args, envir);
    }
}
```

Zrzut ekranu 2

Jeśli znajdujemy się w procesie macierzystym (**fork()** == 1) wykonujemy program tworząc kolejne procesy palaczy. Jeśli jesteśmy w procesie potomnym, wywołujemy program valet.c. W sposób identyczny działa tworzenie palaczy z tą różnicą, że wywołujemy program smoker.c.

3.2 PROCES VALET.C

Odpowiada za proces lokaja. Jego działanie jest bardzo proste.

```
while(1)
{
    sem_wait(sem_valet);
    items = (rand()%3);
    if(items == 0)
    {
        printf("Lokaj: Kładę na stole papier i zapałki.\n");
        sem_post(sem_tobacco);
    }
}
```

Zrzut ekranu 3

Program w nieskończonej pętli (Zrzut ekranu 3) czeka, aż semafor **sem_valet** zostanie zwolniony. Następnie losuje liczbę z przedziału [0,2] i zwalnia semafor odpowiadający wylosowanemu palaczowi.

3.3 PROCES SMOKER.C

Odpowiada za procesy palaczy.

```
int id = atoi(argv[1]);

while(1)
{

    if(id==1)
    {
        sem_wait(sem_tobacco);
        sleep(1);
        printf("Palacz z tytoniem: Dostałem papier i zapałki! Skrecam i pale papierosa\n");
    }
}
```

Zrzut ekranu 4

Zmienna **id** odpowiada za numer identyfikacyjny palacza. W zależności od wartości **id** program będzie czekał aż lokaj położy na stole odpowiednie składniki. Czyli czeka aż zwolni się odpowiedni semafor i wypisze odpowiedni komunikat. Pomiędzy wypisywaniem komunikatów znajduje się funkcja **sleep()**, która ma za zadanie wydłużyć czas pomiędzy kolejnymi palaczami.

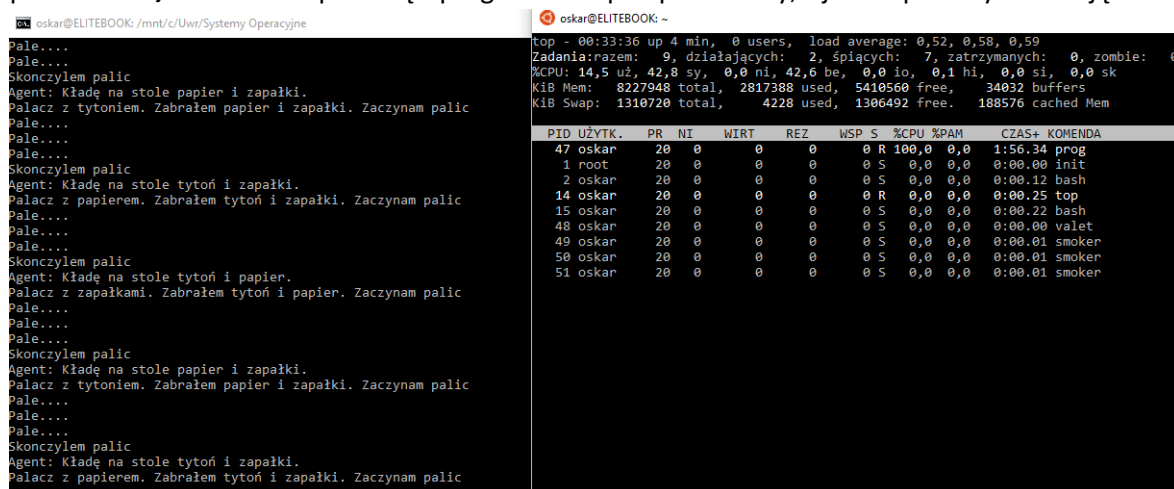
```
sem_post(sem_valet);
```

Zrzut ekranu 5

Na końcu (Zrzut ekranu 5) zwalniamy semafor lokaja umożliwiając mu dalsze układanie składników.

4. TESTOWANIE

Program powinien być testowany w systemie Linux. Przed uruchomieniem należy użyć polecenia **make**, a następnie uruchomić program za pomocą polecenia **./main**. Za pomocą programu **top** sprawdzimy, jakie procesy działają w tle.



```
oskar@ELITEBOOK: /mnt/c/Uwrt/Systemy Operacyjne
Pala....
Pala....
Skonczyłem palic
Agent: Kładę na stole papier i zapałki.
Palacz z tytoniem. Zabrałem papier i zapałki. Zaczynam palic
Pala....
Pala....
Pala....
Skonczyłem palic
Agent: Kładę na stole tytoń i zapałki.
Palacz z papierem. Zabrałem tytoń i zapałki. Zaczynam palic
Pala....
Pala....
Pala....
Skonczyłem palic
Agent: Kładę na stole tytoń i papier.
Palacz z zapałkami. Zabrałem tytoń i papier. Zaczynam palic
Pala....
Pala....
Pala....
Skonczyłem palic
Agent: Kładę na stole papier i zapałki.
Palacz z tytoniem. Zabrałem papier i zapałki. Zaczynam palic
Pala....
Pala....
Pala....
Skonczyłem palic
Agent: Kładę na stole tytoń i zapałki.
Palacz z papierem. Zabrałem tytoń i zapałki. Zaczynam palic
```

```
oskar@ELITEBOOK: ~
top - 00:33:36 up 4 min, 0 users, load average: 0,52, 0,58, 0,59
Zadania:razem: 9, działających: 2, śpiących: 7, zatrzymanych: 0, zombie: 0
%Cpu: 14,5 uż, 42,8 sy, 0,0 ni, 42,6 be, 0,0 io, 0,1 hi, 0,0 si, 0,0 sk
KiB Mem: 8227948 total, 2817388 used, 5410560 free, 34032 buffers
KiB Swap: 1310720 total, 4228 used, 1306492 free, 188576 cached Mem

  PID UŻYTK.  PR  NI   WIRT  REZ   WSP S  %CPU %PAM  CZAS+ KOMENDA
  47 oskar   20   0     0    0    0 R 100,0 0,0  1:56.34 prog
    1 root    20   0     0    0    0 S   0,0 0,0  0:00.00 init
    2 oskar   20   0     0    0    0 S   0,0 0,0  0:00.12 bash
   14 oskar   20   0     0    0    0 R   0,0 0,0  0:00.25 top
   15 oskar   20   0     0    0    0 S   0,0 0,0  0:00.22 bash
   48 oskar   20   0     0    0    0 S   0,0 0,0  0:00.00 valet
   49 oskar   20   0     0    0    0 S   0,0 0,0  0:00.01 smoker
   50 oskar   20   0     0    0    0 S   0,0 0,0  0:00.01 smoker
   51 oskar   20   0     0    0    0 S   0,0 0,0  0:00.01 smoker
```

Zrzut ekranu 6

Program działa poprawnie. Tworzy 4 procesy a komunikaty wyświetlają się poprawnie. Nie zachodzi sytuacja, w której lokaj wyklada składniki przed zakończeniem palenia albo palacz zaczyna palić bez wszystkich składników.