

Systemy operacyjne

Synchronizacja procesów

Kamil Woś

15 stycznia 2018

Spis treści

1. Opis zadania	3
1.1. Treść zadania	3
1.2. Sposób rozwiązania	3
2. Program	4
2.1. Proces dostawcy	4
2.2. Proces pracownika	5
3. Testy	6
4. Literatura	7

1. Opis zadania

1.1. Treść zadania

Fabryka posiada jedno stanowisko produkcyjne. Na stanowisku składane są wyroby z podzespołów X, Y i Z. Podzespoły przechowywane są w magazynie o pojemności M jednostek. Podzespół X zajmuje jedną jednostkę magazynową, podzespół Y dwie, a podzespół Z trzy jednostki. Podzespoły pobierane są z magazynu, przenoszone na stanowisko produkcyjne i montowane. Z podzespołów X, Y i Z po ich połączeniu powstaje jeden produkt, po czym pobierane są następne podzespoły z magazynu. Jednocześnie trwają dostawy podzespołów do fabryki. Podzespoły pochodzą z trzech niezależnych źródeł i dostarczane są w nieokreślonych momentach czasowych.

Napisz program dla procesów dostawca i pracownik, który umożliwi płynną pracę fabryki.

1.2. Sposób rozwiązania

Do rozwiązania tego problemu użyłem dwóch semaforów:

- dostawca,
- stanowisko.

Dostawca dostarcza losowy podzespół, jeśli wszystkie są dostępne w magazynie, w przeciwnym przypadku dostarcza brakujący. W przypadku braku dowolnego z podzespołów w magazynie stanowisko wstrzymuje produkcję.

2. Program

2.1. Proces dostawcy

```
void dostawca()
{
    while(1)
    {
        sem_wait(&zasob->stanowisko);

        przedmiot = rand()%3+1;

        if(zasob->podzespol_x == 0)
            przedmiot = 1;
        else if(zasob->podzespol_y == 0)
            przedmiot = 2;
        else if(zasob->podzespol_z == 0)
            przedmiot = 3;

        pojemnosc = zasob->podzespol_x + (zasob->podzespol_y * 2) + (zasob->podzespol_z * 3);

        if(przedmiot == 1)
        {
            int ile = M - pojemnosc;
            if(ile >= 3)
                zasob->podzespol_x += 3;
            else
                zasob->podzespol_x += ile;
            printf("Dostarczyłem podzespół X! W magazynie znajduje się %d sztuk podzespołów X\n", zasob->podzespol_x);
        }
        else if(przedmiot == 2)
        {
            int ile = (int)((M-pojemnosc)/2);
            if(ile >= 3)
                zasob->podzespol_y += 3;
            else
                zasob->podzespol_y += ile;
            printf("Dostarczyłem podzespół Y! W magazynie znajduje się %d sztuk podzespołów Y\n", zasob->podzespol_y);
        }
        else if(przedmiot == 3)
        {
            int ile = (int)((M-pojemnosc)/3);
            if(ile >= 3)
                zasob->podzespol_z += 3;
            else
                zasob->podzespol_z += ile;
            printf("Dostarczyłem podzespół Z! W magazynie znajduje się %d sztuk podzespołów Z\n", zasob->podzespol_z);
        }

        sem_post(&zasob->dostawca);
    }
}
```

Rys 2.1 Kod procesu dostawcy

Dostawca opuszcza semafor producenta. Następnie losuje, który z podzespołów dostarczy. Po upewnieniu się, że żadnego z podzespołów nie brakuje w magazynie dostarczane są maksymalnie trzy sztuki odpowiedniego przedmiotu. Dokładny kod znajduje się na rysunku 2.1.

2.2. Proces pracownika

```
void pracownik()
{
    while(1)
    {
        sem_wait(&zasob->dostawca);
        if(zasob->podzespol_x >= 1 && zasob->podzespol_y >= 1 && zasob->podzespol_z >= 1)
        {
            if(zasob->przerwa > 0)
            {
                printf("Wznawiam pracę po %d minutach\n", zasob->przerwa);
                zasob->przerwa=0;
            }
            (zasob->podzespol_x)--;
            (zasob->podzespol_y)--;
            (zasob->podzespol_z)--;
            (zasob->wyprodukowane)++;
            printf("Wyprodukowałem jeden przedmiot, łącznie %d sztuk\n", zasob->wyprodukowane);
        }
        else
        {
            printf("Brakuje podzespołów!\n");
            (zasob->przerwa)++;
        }
        sem_post(&zasob->stanowisko);
    }
}
```

Rys 2.2 Kod procesu pracownika

Pracownik opuszcza semafor dostawcy. Następnie sprawdza, czy wszystkie potrzebne podzespoły są dostępne w magazynie. Jeśli tak, tworzy nowy przedmiot. Dodatkowo, w przypadku wznowienia pracy, pracownik informuje o czasie, w którym produkcja była wstrzymana. Dokładny kod znajduje się na rysunku 2.2.

3. Testy

Do uruchomienia programu wymagany jest system Linux. Aby skompilować program należy użyć komendy:

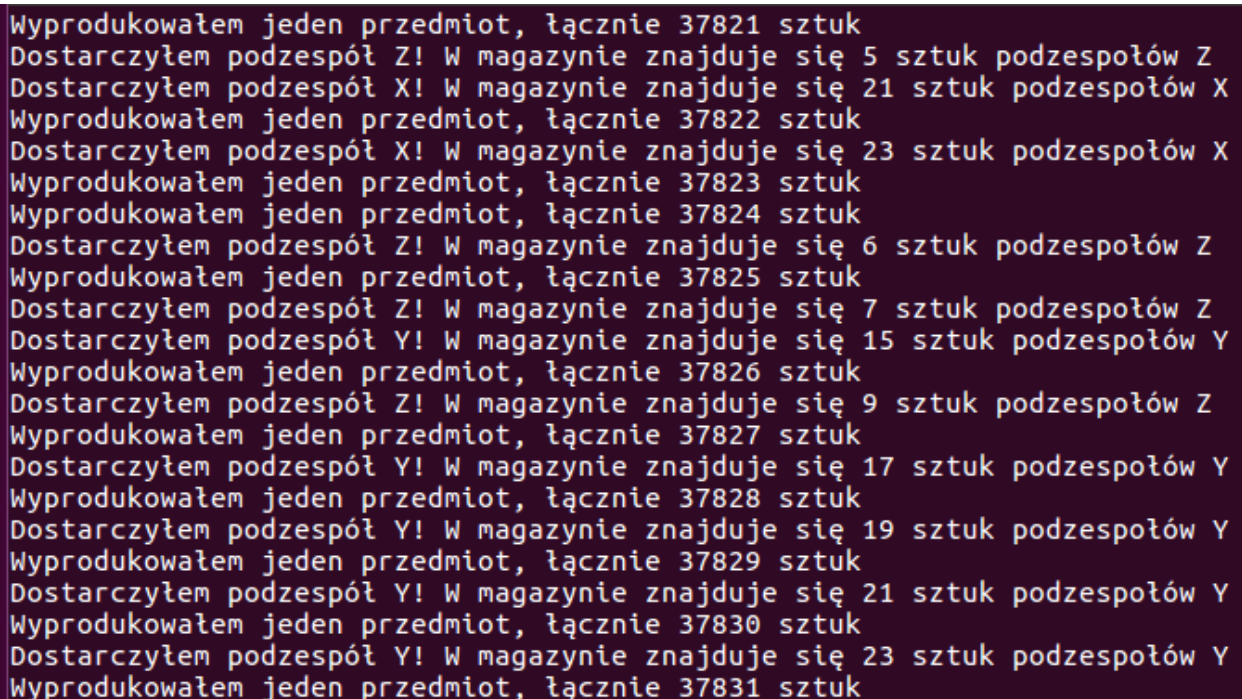
```
gcc -o sem main.c -pthread
```

Program uruchamiamy poleceniem

```
./sem
```

Program uruchamiany był wielokrotnie, zauważając przy tym płynność w działaniu i brak opóźnień.

Przykładowe działanie można zobaczyć na rysunku 3.



```
Wyprodukowałem jeden przedmiot, łącznie 37821 sztuk
Dostarczyłem podzespół Z! W magazynie znajduje się 5 sztuk podzespołów Z
Dostarczyłem podzespół X! W magazynie znajduje się 21 sztuk podzespołów X
Wyprodukowałem jeden przedmiot, łącznie 37822 sztuk
Dostarczyłem podzespół X! W magazynie znajduje się 23 sztuk podzespołów X
Wyprodukowałem jeden przedmiot, łącznie 37823 sztuk
Wyprodukowałem jeden przedmiot, łącznie 37824 sztuk
Dostarczyłem podzespół Z! W magazynie znajduje się 6 sztuk podzespołów Z
Wyprodukowałem jeden przedmiot, łącznie 37825 sztuk
Dostarczyłem podzespół Z! W magazynie znajduje się 7 sztuk podzespołów Z
Dostarczyłem podzespół Y! W magazynie znajduje się 15 sztuk podzespołów Y
Wyprodukowałem jeden przedmiot, łącznie 37826 sztuk
Dostarczyłem podzespół Z! W magazynie znajduje się 9 sztuk podzespołów Z
Wyprodukowałem jeden przedmiot, łącznie 37827 sztuk
Dostarczyłem podzespół Y! W magazynie znajduje się 17 sztuk podzespołów Y
Wyprodukowałem jeden przedmiot, łącznie 37828 sztuk
Dostarczyłem podzespół Y! W magazynie znajduje się 19 sztuk podzespołów Y
Wyprodukowałem jeden przedmiot, łącznie 37829 sztuk
Dostarczyłem podzespół Y! W magazynie znajduje się 21 sztuk podzespołów Y
Wyprodukowałem jeden przedmiot, łącznie 37830 sztuk
Dostarczyłem podzespół Y! W magazynie znajduje się 23 sztuk podzespołów Y
Wyprodukowałem jeden przedmiot, łącznie 37831 sztuk
```

Rys. 3 Przykładowe działanie programu

4. Literatura

- http://wazniak.mimuw.edu.pl/index.php?title=SOP_lab_nr_12