

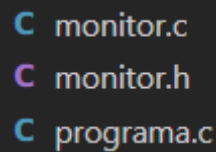
SSC0640 – Sistemas Operacionais 1 – Relatório

Marcos Vinícius Firmino Pietrucci

10770072

1- Implementação

O programa foi implementado em linguagem C. Para tal, separei meu programa em três arquivos.



C monitor.c
C monitor.h
C programa.c

O arquivo “programa.c” contém a main e as chamadas das threads. Conforme o enunciado, criei 5 threads iguais que se executam em loop e paralelamente, uma para cada filósofo.

O arquivo “monitor.c” contém todo o desenvolvimento do monitor, com suas várias funções. Foi utilizado um semáforo para controlar o acesso aos recursos (palitos), de forma a manter o paralelismo mas evitar o problema de exclusão mútua. O arquivo “monitor.h” é o header do “monitor.c”

De forma geral, as threads são ativadas no “programa.c” e tentam acessar os palitos por meio de funções no “monitor.c”. Essas funções, por sua vez, utilizam o semáforo para coordenar o acesso e impedir deadlocks.

```
//Thread dos filósofos
void *thread_filosofo(void *i)
{
    int indc;
    int tempo;

    //Variavel que armazena indice do filósofo
    indc = * (int*) i;

    while(TRUE)
    {
        //Tempo que o filósofo ficara pensando
        tempo = (1 + rand()) % 10;

        //Imprime mensagem indicativa
        printf("\nFilosofo %d pensa por %ds", indc, tempo);
        sleep(tempo);

        //Após acordar, o filósofo tentará pegar os palitos
        pegar_palitos(indc);

        //Tempo que o filósofo ficara comendo
        tempo = (1 + rand()) % 10;

        //Imprime mensagem indicativa
        printf("\nFilosofo %d come por %ds", indc, tempo);
        sleep(tempo);
        printf("\nFilosofo %d terminou de comer", indc);

        //O filósofo, após comer, devolve os palitos
        devolve_palitos(indc);
    }

    return NULL;
}
```

2- Resultados obtidos

Os resultados foram positivos. As saídas do programa foram as ações dos filósofos. A cada instante de tempo um filósofo tomava a iniciativa de pensar, ficar com fome e comer. Depois, o filósofo pausa por um instante de tempo.

Ao executar o programa, note que a saída, apesar de desorganizada devido ao paralelismo das threads, comprova o compartilhamento ordenado dos recursos (palitos).

```
marcos@DESKTOP-TTFLT3D:/mnt/c/Users/MarcosViniciusFirmin/Desktop/5º Período/Sistemas Operacionais/Problema-dos-5-filosofos$ make run
./prog.out

Filosofo 1 pensa por 4s
Filosofo 2 pensa por 7s
Filosofo 3 pensa por 8s
Filosofo 4 pensa por 6s
Filosofo 0 pensa por 4s
Filosofo 1 come por 6s
Filosofo 4 come por 7s
Filosofo 1 terminou de comer
Filosofo 1 pensa por 3s
Filosofo 2 come por 0s
Filosofo 2 terminou de comer
Filosofo 2 pensa por 2s
Filosofo 2 come por 3s
Filosofo 4 terminou de comer
Filosofo 4 pensa por 8s
Filosofo 0 come por 1s
Filosofo 0 terminou de comer
Filosofo 0 pensa por 0s
Filosofo 0 come por 4s
Filosofo 2 terminou de comer
Filosofo 2 pensa por 7s
```

3- Problemas encontrados

Meu maior problema foi com a implementação do monitor. Mesmo estudando o conteúdo e dominando o assunto conceitualmente, no momento de pôr em prática eu senti bastante dificuldade.

Comecei a procurar por referências em sites e vídeos, e de pouco em pouco fui construindo o meu monitor.

4- Como executar

Para executar meu programa, pode utilizar o meu Makefile. Para isso, execute no terminal do Linux, dentro do diretório com todos os arquivos, o comando:

```
make prog
```

O comando acima irá compilar os arquivos, gerar os arquivos “.o” e gerar o “prog.out”, arquivo executável. Para executar, use o comando:

```
make run
```

Após estes dois comandos, o programa deve começar a rodar no seu terminal. Caso não queira usar meu makefile, pode executar o programa da maneira clássica, rodando os seguintes comandos de forma sequencial:

```
gcc -c -o monitor.o monitor.c -pthread -Wall  
gcc -c -o programa.o programa.c -pthread -Wall  
gcc -o prog.out monitor.o programa.o -pthread -Wall  
./prog.out
```