USP

SISTEMAS DE INFORMAÇÃO

JANTAR DOS FILÓSOFOS(COM MONITOR)

RELATORIO SOBRE A ATIVIDADE

SSC0541:SISTEMAS OPERACIONAIS

VANDERLEI BONATO

MIRASSOL

2021

JOÃO PAULO GARCIA MARTINELLI DE OLIVEIRA

11816056

Explicação logica

O código está dividido em 3 partes:

A biblioteca monitor.h, onde estão definidos algumas variáveis e funções.

Em monitor.c, foi desenvolvido um monitor genérico usando semáforos com 2 funções principais, ‘wait’ na qual a função joga o filósofo em um de 2 semáforos

(mutex ou próximo), após o wait, temos o ‘signal’, na qual se o contador do filósofo está positivo, ele manda o sinal para a função.

Ainda em monitor.c, começamos a definir as funções para o jantar, a primeira é o testar(), na qual testamos o filósofo, verificando se ele pode comer, após essa, temos a pega\_talher(int i) que altera o estado do filósofo para fome, testa se pode comer e enquanto está com fome ele espera, finalmente, solta\_talher(int i) na qual ele volta a pensar.

Em main.c, o \*jantar(void \*i) é a função que organiza o roteiro do jantar, sendo realizado na main.

Problemas encontrados

Tive dificuldades em começar o exercício, o conceito de monitor ficou meio vago para mim, após algumas video aulas em portugues eu consegui entender e voltei a travar no problema da starvation, eu resolvi usando o esquema de ciclos, e definindo quais filósofos comem no ciclo 1, foi uma resolução mais simples mas acabou funcionando.

Como executar

Abra a pasta no terminal, utilize o comando $Make para compilar

após isso, utilize o comando $./jantar\_executavel para rodar o código

Codigo fonte:

(está disponível também como arquivo)

monitor.h

//Aqui sera feito a declaração das variaveis e da biblioteca de funções

#ifndef monitor\_h

#define monitor\_h

//numero de filosofos, pode ser alterado

//

#define NUMERO\_FILOSOFOS 5

//estados

#define PENSANDO 0

#define FOME 1

#define COMENDO 2

//definindo a estrutura do circulo

#define ESQUERDA (i+NUMERO\_FILOSOFOS-1)%NUMERO\_FILOSOFOS

#define DIREITA (i+1)%NUMERO\_FILOSOFOS

//funcoes que utilizarei

void iniciar();

void testar( int i);

void pegar\_talher(int i);

void soltar\_talher(int i);

#endif

Monitor.c

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

#include <ctype.h>

#include "monitor.h"

sem\_t Mutex;

sem\_t Proximo;

//conta no sem\_t Proximo

int prox\_sinal=0;

struct monitor\_datatype{

sem\_t semaforo;

int contador;

};

typedef struct monitor\_datatype monitor\_data;

monitor\_data x[NUMERO\_FILOSOFOS];

int estado[NUMERO\_FILOSOFOS]; //ja declarado antes

int ciclo[NUMERO\_FILOSOFOS];

//lembrar que as 2 principais funcoes de um monitor sao wait e signal

void wait(int i){

//oque esta acontecendo:

//pego a lista de filosofos e add 1

x[i].contador++;

//se o sinal for positivo, uso o semaforo Proximo

if(prox\_sinal>0) {

sem\_post(&Proximo);

}

//caso contrario, uso o Mutex

else {

sem\_post(&Mutex);

}

//espero o semaforo

sem\_wait(&x[i].semaforo);

//retorno a posicao do contador

x[i].contador--;

}

void signal(int i){

//verificar o contador dentro do monitor

if(x[i].contador>0){

//apos isso subir o sinal

prox\_sinal++;

sem\_post(&x[i].semaforo);

sem\_wait(&Proximo);

//apos realizar as operacoes, liberar o sinal

prox\_sinal--;

}

}

void testar(int i){

if(estado[i]==FOME && estado[ESQUERDA] != COMENDO && estado[DIREITA] != COMENDO && ciclo[i] == i && ciclo[ESQUERDA] == i){

estado[i]=COMENDO;

//mando o sinal que este esta comendo

signal(i);

}

}

void pegar\_talher(int i){

//segurar o mutex

sem\_wait(&Mutex);

//mudo o estado

estado[i]=FOME;

testar(i);

//jogo o filosofo no teste para ver se ele pode comer

while(estado[i]==FOME){

//aqui espera ate liberar

wait(i);

}

if(prox\_sinal>0){

//se tiver sinal, joga para o proximo semafoto

sem\_post(&Proximo);

}

else{

//cc, Mutex

sem\_post(&Mutex);

}

}

void soltar\_talher(int i){

//entro no estado critico

sem\_wait(&Mutex);

estado[i]=PENSANDO;

//mudar o ciclo

//Avanço em 1 unidade na mesa

ciclo[i]=DIREITA;

//E coloco o ciclo anterior apontando para a esqueda do prox;

ciclo[ESQUERDA]=ESQUERDA;

//apos isso testar a posicao

testar(ESQUERDA);

testar(DIREITA);

if(prox\_sinal>0){

//se tiver sinal, joga para o proximo semafoto

sem\_post(&Proximo);

}

else{

//cc, Mutex

sem\_post(&Mutex);

}

}

void iniciar(){

int i;

//iniciar os semaforos

sem\_init(&Mutex,0,1);

sem\_init(&Proximo,0,0);

//setar variaveis

for(i=0;i<NUMERO\_FILOSOFOS;i++){

estado[i]=PENSANDO;

sem\_init(&x[i].semaforo,0,0);

x[i].contador =0;

ciclo[i]=i;

}

ciclo[1] = 2;

ciclo[3] = 4;

ciclo[6] = 0;

}

Main.c

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#include <string.h>

#include <stdlib.h>

#include <ctype.h>

#include "monitor.h"

void \*jantar(void \*i){

while(1){

//declarar filosofo

int atual=\*(int \*)i;

int n,m;

n=rand();

n=n%10;

printf("\n Filosofo %d esta pensando por %d s",atual,n);

//segurar os n seg

sleep(n);

pegar\_talher(atual);

m= rand();

m= m % 7;

printf("\n Filosofo %d esta comendo por %d s",atual,m);

sleep(m);

soltar\_talher(atual);

}

}

int main()

{

int i, pos[NUMERO\_FILOSOFOS];

pthread\_t thread[NUMERO\_FILOSOFOS];

pthread\_attr\_t attr;

iniciar();

pthread\_attr\_init(&attr);

for (i = 0; i < NUMERO\_FILOSOFOS; i++)

{

pos[i] = i;

//create thread corresponding to each philosopher

pthread\_create(&thread[i], NULL,jantar, (int \*) &pos[i]);

}

for (i = 0; i < NUMERO\_FILOSOFOS; i++)

{

pthread\_join(thread[i], NULL);

}

return 0;

}

Resultados

Os resultados obtidos ao rodar o código, existe um delay de alguns segundos antes do programa começar a funcionar, não consegui reproduzir em outros computadores para averiguar.

Filosofo 1 esta pensando por 6 s

Filosofo 3 esta pensando por 5 s

Filosofo 0 esta pensando por 3 s

Filosofo 2 esta pensando por 7 s

Filosofo 4 esta pensando por 3 s

Filosofo 4 esta comendo por 3 s

Filosofo 0 esta comendo por 3 s

Filosofo 4 esta pensando por 2 s

Filosofo 2 esta comendo por 1 s

Filosofo 3 esta comendo por 3 s

Filosofo 2 esta pensando por 2 s

Filosofo 1 esta comendo por 5 s

Filosofo 0 esta pensando por 0 s

Filosofo 4 esta comendo por 4 s

Filosofo 3 esta pensando por 3 s

Filosofo 2 esta comendo por 0 s

Filosofo 1 esta pensando por 0 s

Filosofo 2 esta pensando por 6 s

Filosofo 3 esta comendo por 2 s

Filosofo 0 esta comendo por 2 s

Filosofo 4 esta pensando por 1 s

Filosofo 3 esta pensando por 8 s

Filosofo 4 esta comendo por 5 s

Filosofo 1 esta comendo por 2 s

Filosofo 0 esta pensando por 2 s

Filosofo 1 esta pensando por 0 s

Filosofo 2 esta comendo por 5 s

Filosofo 0 esta comendo por 6 s

Filosofo 4 esta pensando por 7 s

Filosofo 2 esta pensando por 5 s

Filosofo 3 esta comendo por 5 s

Filosofo 1 esta comendo por 0 s

Filosofo 0 esta pensando por 2 s