

**Escola Superior de Tecnologia de Tomar**

Licenciatura em Engenharia Informática

Sistemas Operativos

**RELATÓRIO DO TRABALHO PRÁTICO**

THREADS

## Andreia Maria Rodrigues Ferreira

## Nº 19985

Índice

**Introdução3**

**Código Inicialmente Utilizado4**

**Código Utilizado7**

**Fluxogramas11**

Fluxograma Thread Produtora11

Fluxograma Thread Consumidora12

Fluxograma Main13

**Conclusão14**

Introdução

Este trabalho prático, da cadeira de Sistemas Operativos, tem como objetivo introduzir-nos conhecimentos relativos a Threads. Threads são a unidade básica de utilização do CPU e com a sua utilização, é possível que um mecanismo em que processos sequenciais efetuem chamadas ao sistema de bloqueio e mantenha o funcionamento em paralelo. No ramo da informática podemos dizer que as Threads estão muitas vezes presentes sem nos darmos conta disso, como por exemplo em jogos.

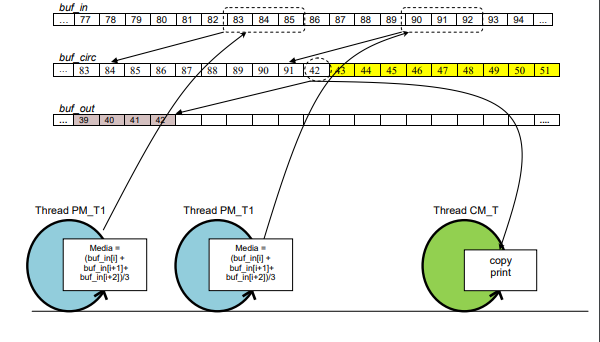
Neste trabalho irão ser usados 3 buffers que contém números inteiros:

* O primeiro buffer, será designado por **buffer\_in**, contém 30000 elementos ordenados, ou seja, os valores de 1 a 30000.
* O segundo buffer, designado por **buffer\_circ**, suporta até 50 elementos.
* E por fim, o terceiro buffer, **buffer\_out**, suporta até 29998 elementos.

Para além dos buffers, irão ser usadas 3 threads, designadas por PM\_T1 e PM\_T2 que são threads produtoras, e CM\_T que é uma thread consumidora. Têm como funções:

* As threads PM\_1 e PM\_2, preenchem em simultâneo o buffer\_circ, com o resultado da média de 3 elementos do buffer\_in.
* A thread CM\_T, percorre o buffer\_circ e copier esses elementos para o buffer\_out.

Aqui se encontra um exemplo da situação que pode ocorrer:



Código Inicialmente Utilizado

No início de desenvolvimento do trabalho, decidi desenvolver funções em linguagem c, para perceber qual será a melhor forma da realização do mesmo. Aqui se encontra o respetivo código:

#include <stdio.h>

#include <malloc.h>

#include <pthread.h>

//criação dos buffers representados em arrays

int[30000] buffer\_in;

int[50] buffer\_circ;

int[29998] buffer\_out;

//variaveis auxiliares

int bufInAux1 = 0;

int bufInAux2 = 0;

int bufCircAux = 0;

//função que permite inserir os numeros de 1 a 30000 no buffer\_in

void insereValoresBufferIn(){

    int i;

int tam = sizeof(buffer\_in)/sizeof(buffer\_in[0]);

//printf("Valores do buffer");

    for(i=1;i<=tam;i++){

        buffer\_in[i]=i;

tam++;

        //printf("|%d",buffer\_in[i]);

    }

}

//funcao que insere um valor no buffer\_circ

void insereValoresBufferCirc(int valor){

int i;

int tam = sizeof(buffer\_circ)/sizeof(buffer\_circ[0]);

for(i=0; i<tam; i++){

buffer\_circ[i]=valor;

}

}

//funcao que insere valores no buffer\_out

void insereValoresBufferOut(int valor){

int i;

int tam = sizeof(buffer\_out)/sizeof(buffer\_out[0]);

for(i=0; i<tam; i++){

buffer\_out[i]=valor;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Threads Produtoras\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//1ª Thread

void\* PM\_T1(){

while(1){

//calcula a media

int media = ((buffer\_in[bufInAux1] + buffer\_in[bufInAux1+1] + buffer\_in[bufInAux1+2])/3);

//insere o seu valor no array buffer\_out

insereValoresBufferCirc(media);

//incrementa a variavel bufInAux

bufInAux = bufInAux + 1;

}

}

//2ª Thread

void\* PM\_T2(){

while(1){

//calcula a media

int media = ((buffer\_in[bufOutAux2] + buffer\_in[bufOutAux2+1] + buffer\_in[bufOutAux2+2])/3);

//insere o seu valor no array buffer\_out

insereValoresBufferCirc(media);

//incrementa a variavel bufOutAux

bufOutAux = bufOutAux + 1;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Threads Consumidoras\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//esta thread tem de ir ao buffer\_circ , retirar um valor e colocar no buffer\_out

//quando o buffer\_circ esta cheio, esta thread tem de adormecer as restantes, e acorda as quando ja existir espaço

//disponivel no buffer

//para verificar se o buffer esta vazio é com um while

void\* CM\_T(){

while(1){

//retirar valor do buffer\_circ

int i;

int tam = sizeof(buffer\_circ)/sizeof(buffer\_circ[0]);

for(i=1;i<=tam;i++){

int num = buffer\_circ[i];

buffer\_circ[i]=0;

insereValoresBufferOut(num);

}

}

}

}

int main(){

//criação das threads

pthread\_create (PM\_T1, NULL, , NULL);

pthread\_create (PM\_T2, NULL, , NULL);

pthread\_create(CM\_T, NULL, ; NULL);

}

Código Utilizado

Após perceber melhor a lógica do trabalho realizei o meu próprio código, que é o seguinte:

//bibliotecas necessarias para a execucao do trabalho

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include <pthread.h>

//variaveis

#define VALORESIN 82

#define VALORESCIRC 30

#define VALORESOUT 80

//buffers necessarios para a execução do trabalho

int buffer\_in[VALORESIN];

int buffer\_circ[VALORESCIRC];

int buffer\_out[VALORESOUT];

//função que permite inserir valores no buffer\_in

void insereValoresBufferIn(){

int i;

    for(i=1;i<=VALORESIN;i++){

        buffer\_in[i-1]=i;

    }

}

//estrutura para a thread produtora

struct{

pthread\_mutex\_t mutex;

pthread\_cond\_t cond;

//iterador que indica a posicao no buffer\_in

int nput;

int iteradorThread;

} produtora={PTHREAD\_MUTEX\_INITIALIZER, PTHREAD\_COND\_INITIALIZER};

//estrutura para a thread consumidora

struct{

pthread\_mutex\_t mutex;

pthread\_cond\_t cond;

//permite quantos valores se encontram na thread buffer\_circ

int nready;

int iterador;

int iteradorOut;

} consumidora={PTHREAD\_MUTEX\_INITIALIZER, PTHREAD\_COND\_INITIALIZER};

//função pertencente a thread produtora

void \* produzir(void\* arg){

//ciclo infinito

while(1){

//verificar se se pode colocar no buffer\_in

if(consumidora.nready<VALORESCIRC){

buffer\_circ[produtora.iteradorThread] = ((buffer\_in[produtora.nput] + buffer\_in[produtora.nput + 1] + buffer\_in[produtora.nput + 2])/3);

\*((int \*)arg) += 1;

if(produtora.iteradorThread>=VALORESCIRC-1){

produtora.iteradorThread=0;

}

else{

produtora.iteradorThread=produtora.iteradorThread+1;

}

produtora.nput = produtora.nput + 1;

//bloqueia todas as consumidoras, pois so pode "atuar" esta

pthread\_mutex\_lock(&consumidora.mutex);

//incrementa o numero de dados preparados

consumidora.nready=consumidora.nready + 1;

pthread\_cond\_signal(&consumidora.cond);

//thread consumidora desbloqueada

pthread\_mutex\_unlock(&consumidora.mutex);

}

else{

//produtora é adormecida

pthread\_mutex\_unlock(&produtora.mutex);

pthread\_cond\_wait(&produtora.cond, &produtora.mutex);

}

//condicao de paragem, ou seja zona critica

if(produtora.nput >= VALORESOUT){

//desbloqueia todas as produtoras

pthread\_mutex\_unlock(&produtora.mutex);

//termina a threads

return(NULL);

}

}

}

void \* consumir( void\* arg){

int i;

while(1){

if(consumidora.iteradorOut>=VALORESOUT){

return(NULL);

}

//verifica se ha valores a retirar do buffer\_circ

while(consumidora.nready==0){

//adormece a thread pois ela não tem nada que fazer

pthread\_cond\_wait(&consumidora.cond, &consumidora.mutex);

}

//coloca valores no buffer\_circ

buffer\_out[consumidora.iteradorOut]=buffer\_circ[consumidora.iterador];

\*((int \*)arg) += 1;

//como ja retirou um valor do buffer\_circ acordar as produtoras

if(consumidora.nready<VALORESOUT){

pthread\_cond\_signal(&produtora.cond);

}

//quando chega ao fim do buffer\_circ volta ao inicio

if(consumidora.iterador>=VALORESCIRC-1){

consumidora.iterador=0;

}

else{

consumidora.iterador=consumidora.iterador + 1;

}

consumidora.iteradorOut=consumidora.iteradorOut + 1;

//desincrementar nready

consumidora.nready=consumidora.nready - 1;

}

}

int main(){

int var1=0;

int var2=0;

int var3=0;

insereValoresBufferIn();

//inicializacao da thread das threads

pthread\_t PM\_T1, PM\_T2, CM\_T;

//criação das threads

pthread\_create(&PM\_T1, NULL, produzir, &var1);

pthread\_create(&PM\_T2, NULL, produzir, &var2);

pthread\_create(&CM\_T, NULL, consumir, &var3);

//espera que a thread finalize

pthread\_join(PM\_T1, NULL);

pthread\_join(PM\_T2, NULL);

pthread\_join(CM\_T, NULL);

int i;

printf("Buffer\_IN: \n");

for(i=0; i<VALORESIN; i++){

printf("|%d", buffer\_in[i]);

}

printf("\n");

printf("Buffer\_CIRC: \n");

for(i=0; i<VALORESCIRC; i++){

printf("|%d", buffer\_circ[i]);

}

printf("\n");

printf("Buffer\_OUT: \n");

for(i=0; i<VALORESOUT; i++){

printf("|%d", buffer\_out[i]);

}

printf("\n");

printf("Numero de vezes que a thread executou: %d \n", var1);

printf("Numero de vezes que a thread executou: %d \n", var2);

printf("Numero de vezes que a thread executou: %d \n", var3);

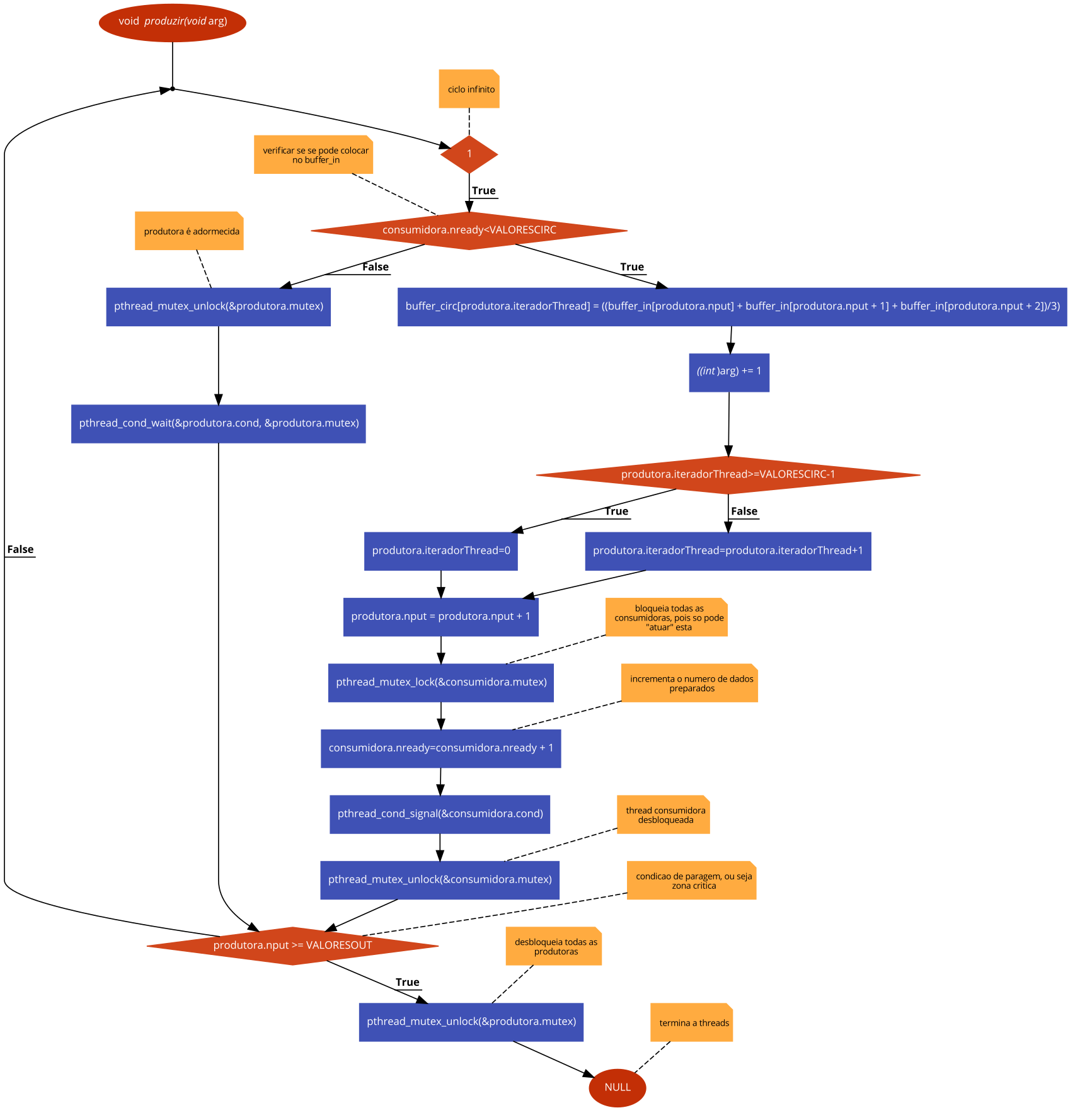
exit(0);

}

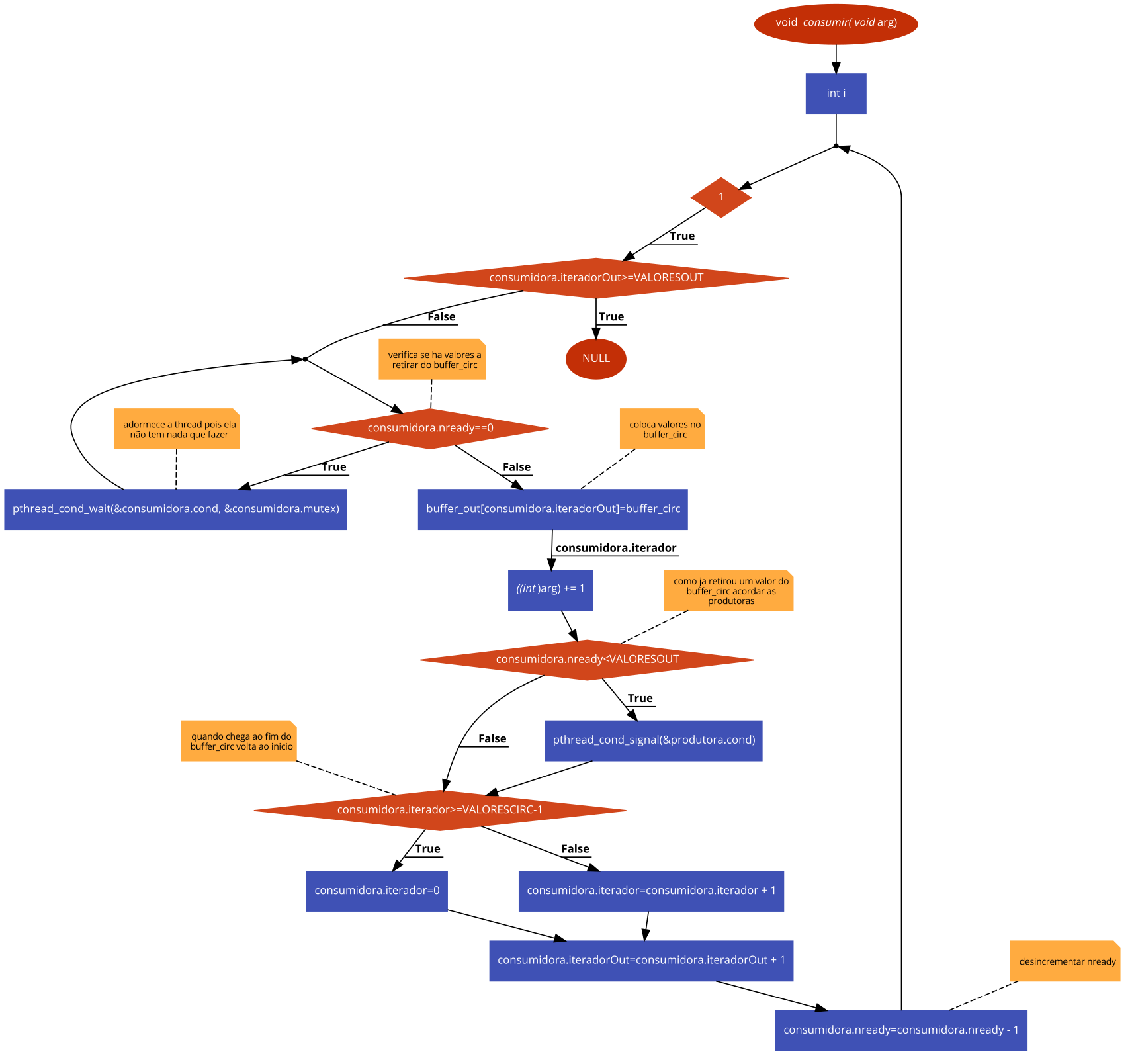
Fluxogramas

Aqui estão representados os fluxogramas das threads utilizadas no trabalho:

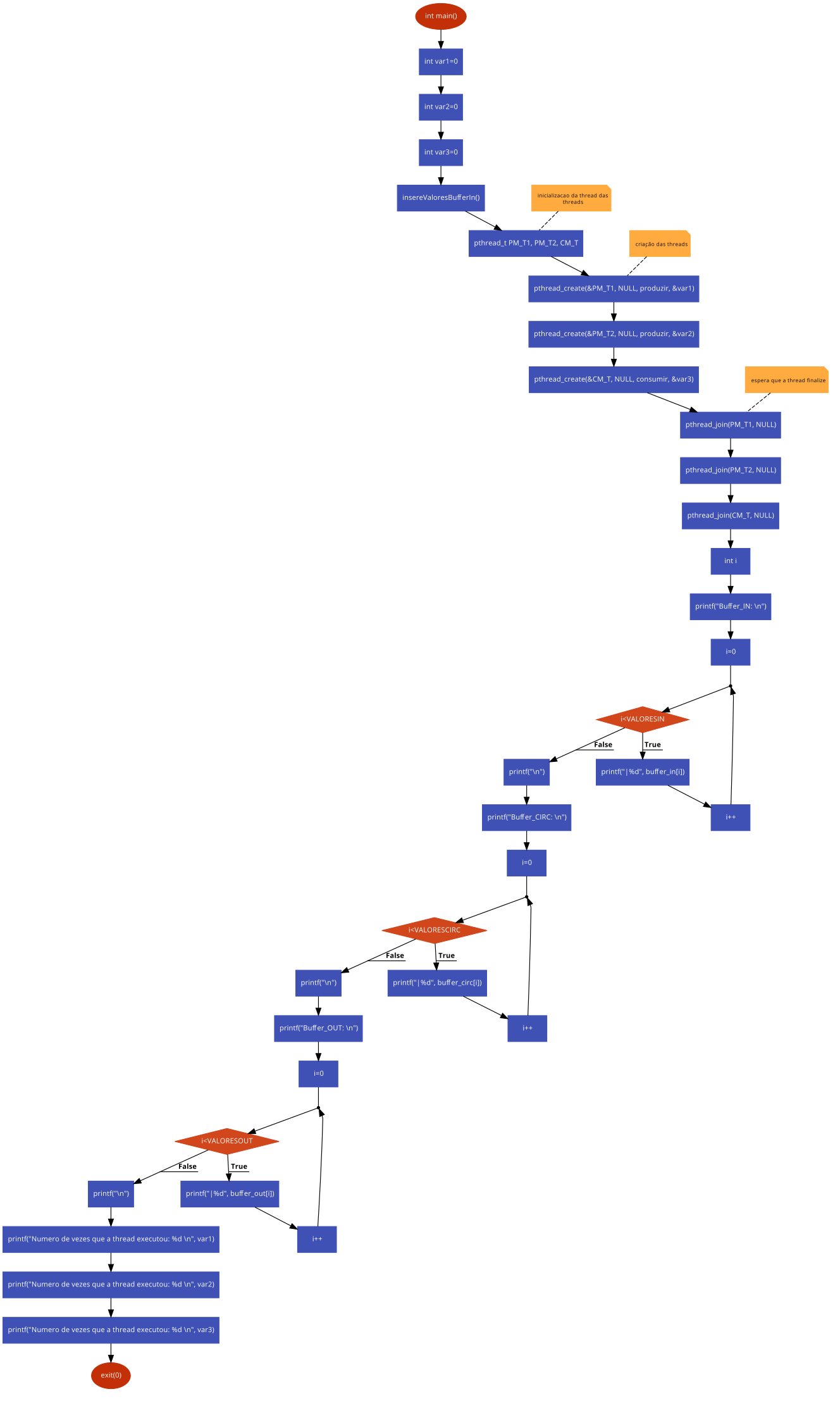
* Fluxograma Thread Produtora:



* Fluxograma Thread Consumidora



* Fluxograma Main



Conclusão

Este trabalho, apesar de não ter sido fácil de realizar, foi uma boa ajuda para perceber o que são threads e como as mesmas funcionam, visto que têm de funcionar todas em simultâneo, e é algo que acaba por estar presente diariamente na nossa área. Para além das theads, aprendi a usar buffer’s que acabam por ser idênticos a array’s. Tenho de realçar também que este trabalho foi uma mais valia para rever a linguagem de programação c, visto que já à algum tempo não programava em c.