Universidade Federal de Santa Catarina - UFSC
Bacharelado em Sistemas de Informação

INE5611-04238A/B (20212) - Sistemas Operacionais

Entrega T.1

Grupo 6 - participantes:

- Ayron Scartezini do Nascimento (18105484)
- Beatriz Valio Weiss (21205057)

Escolhas de projeto

- Estrutura: pilha (*Entregues* e *Disponíveis*)
    - Pilha inicia apontando para null;
    - Remoção sempre do topo da pilha;
    - Adição sempre no topo da pilha;
    - Disponibilização esvazia a pilha de Entregues e passa para o topo da pilha de Disponíveis;
    - Aluguel remove do topo da pilha de disponíveis
    - Entrega adiciona ao topo da pilha de Entregues
- Debugar com o uso de *printf()* e funções que imprimem os imóveis nas pilhas, uma vez que não conseguimos debugar normalmente o código em linguagem C usando Visual Studio Code no windows, com WSL.

Descrição Código

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <unistd.h>
```

Parte inicial do código com as bibliotecas necessárias.

```c
char endereco[5][30] = {"R. Conselheiro Mafra, 123", "R. João Marçal, 234", "R. Maria Eduarda, 345", "R. João Vieira, 456", "R. Tangara, 567"};
char preco[5][20] = {"RS 1250,00", "RS 1999,99", "RS 2500,00", "RS 2760,30", "RS 2040,00"};
```

```
char bairro[5][10] = {"Centro", "Trindade", "Pantanal", "Capoeiras",
"Agronomica"};

typedef struct Imovel {
 char codigo[5];
 char endereco[50];
 char preco[15];
 char bairro[30];

struct Imovel *anterior;
} Imovel;
```

As variáveis das informações dos imóveis são declaradas (*endereço, preço, bairro*) como arrays bidimensionais contendo os valores possíveis para a criação de um novo imóvel (5 para cada variável).

A estrutura de imóvel é declarada com os atributos *código, endereço, preço* e *bairro* - sendo estes um array de char com tamanho suficiente para os valores das variáveis anteriormente declaradas. Código é incrementado a cada criação de imóvel, dessa forma evitando que dois imóveis possuam o mesmo código e, dado que o tamanho máximo de caracteres do código é 5, incluindo o char '#', podem existir até 9999 imóveis únicos.

É implementado também a estrutura para o ponteiro anterior, que será útil para a realização do sistema em estrutura de pilha.

```
Imovel *topoDisponiveis = NULL;
Imovel *fundoDisponiveis = NULL;

Imovel *topoEntregues = NULL;
Imovel *fundoEntregues = NULL;

pthread_mutex_t mutexInc;
pthread_mutex_t mutexPilhas;

int inc;
```

Há a implementação dos ponteiros das pilhas.

Pilha 1 - Imóveis Disponíveis: contando com o ponteiro de topo (último da pilha) e fundo (primeiro da pilha).

Pilha 2 - Imóveis Entregues: contando com o ponteiro de topo (último da pilha) e fundo (primeiro da pilha).

Além disso, o valor de incremento para o atributo *código* será aplicado a partir da variável inteiro *inc*.

Atenção especial para o fato da **zona crítica do sistema** estar neste trecho. Isto porque as variáveis são globais, sendo sua utilização passível de utilização simultânea - através das threads - nas funções que as utilizam caso não houvesse a utilização de semáforo ou mutex. No entanto, **implementamos os mutex** *mutexInc* para o inteiro *inc* e *mutexPilhas* para a manipulação dos ponteiros utilizados nas pilhas - o que no decorrer do código impedirá a utilização simultânea das variáveis globais enquanto em execução.

Como utilizamos o mutex

O mutex - exclusão mútua - é uma técnica para evitar que duas ou mais threads tenham acesso simultaneamente a um recurso compartilhado, no nosso caso as variáveis globais mencionadas anteriormente, acesso esse conhecido como zona crítica.

A lógica dessa técnica é sempre a mesma: ao iniciar uma função que utilize a zona crítica, há o bloqueio da utilização da zona crítica. Ao final da utilização da zona crítica, o mutex a desbloqueia, liberando então que outra thread faça essa utilização.

Essa técnica foi utilizada em todas as ações executadas tanto por inquilino quanto por corretor e, além disso, foi utilizada também na função de popular imóvel, a qual explicará melhor o mutex a seguir.

```
void popularImovel(struct Imovel *imovel) {
 pthread_mutex_lock(&mutexInc);
 sprintf(imovel->codigo, "#%d", inc++);
 pthread_mutex_unlock(&mutexInc);
 sprintf(imovel->endereco, "%s", endereco[rand() % 5]);
 sprintf(imovel->preco, "%s", preco[rand() % 5]);
 sprintf(imovel->bairro, "%s", bairro[rand() % 5]);
}
```

A função *popularImovel* é responsável por fazer a atribuição dos valores randomizados para os atributos de um novo imóvel.

O mutex *mutexInc* bloqueia o acesso a variável inc para que caso dois imóveis estejam sendo criados concorrentemente, por exemplo um primeiro imovel não tenha um codigo incrementado pelo segundo imovel, o que causaria perdas de possíveis associações de código, diminuindo assim a quantidade limite de imóveis únicos.

```
void criarImovel(int id)
{
 Imovel *novoTopoDisponivel = (struct Imovel *) malloc(sizeof(struct
Imovel));
 popularImovel(novoTopoDisponivel);

 pthread_mutex_lock(&mutexPilhas);
```

```
 novoTopoDisponivel->anterior = topoDisponiveis;
 topoDisponiveis = novoTopoDisponivel;
 if(fundoDisponiveis == NULL) {
   fundoDisponiveis = topoDisponiveis;
 }

 pthread_mutex_unlock(&mutexPilhas);

 printf("CORRETOR %d adicionou Imóvel %s\n", id,
novoTopoDisponivel->codigo);
 printf("-----------------------------------\n");
 printf(" Codigo: %s\n"
        " Endereço: %s\n"
        " Preço: %s\n"
        " Bairro: %s\n",
        novoTopoDisponivel->codigo,
        novoTopoDisponivel->endereco,
        novoTopoDisponivel->preco,
        novoTopoDisponivel->bairro);
 printf("-----------------------------------\n");
};
```

A função *criarImovel* é responsável pela criação de um novo imóvel a partir da chamada da função *popularImovel* e alimentação da pilha de imóveis disponíveis - de fato disponibilizando o imóvel. Dessa forma, caso a pilha esteja vazia, o fundo passa a ser o primeiro imóvel adicionado. Por fim então é impresso as características deste imóvel - valores estes randomizados em *popularImovel*.

Atenção para a utilização de *malloc* para alocar endereço de memória na criação do imóvel, garantindo que o endereço desse objeto esteja acessível ao programa - prevenindo *segmentation fault*.

```
void removerImovel(int id) {
 pthread_mutex_lock(&mutexPilhas);
 if(topoDisponiveis != NULL) {
   printf("CORRETOR %d removeu imovel %s\n", id, topoDisponiveis->codigo);
   if(fundoDisponiveis == topoDisponiveis) {
     topoDisponiveis = NULL;
     fundoDisponiveis = NULL;
   } else {
     Imovel *temp = topoDisponiveis;
     topoDisponiveis = topoDisponiveis->anterior;
```

```
      free(temp);
    }
  } else {
    printf("CORRETOR %d tentou remover imovel porem nao ha imoveis
disponiveis\n", id);
  }
    pthread_mutex_unlock(&mutexPilhas);
}
```

A função *removerImovel* é responsável por retirar da pilha de imóveis disponíveis o imóvel que, por decisão do corretor, não será mais possibilitado de ser alugado.

Caso tenha imóveis disponíveis, há a verificação se é o único ou não para que os ponteiros sejam corretamente manipulados e por fim é impresso o *código* do imóvel removido.

Caso não tenha, é impresso que houve essa tentativa mas sem êxito.

```
void alugarImovel(int id, Imovel **imovel){
  if((*imovel) == NULL) {
    pthread_mutex_lock(&mutexPilhas);
    if(topoDisponiveis != NULL) {
      (*imovel) = topoDisponiveis;
      topoDisponiveis = topoDisponiveis->anterior;
      (*imovel)->anterior = NULL;
      if(topoDisponiveis == NULL) {
        fundoDisponiveis == NULL;
      }
      printf("Inquilino %d alugou imovel %s\n", id, (*imovel)->codigo);
    } else {
      printf("Inquilino %d tentou alugar imovel porem nao ha imovel
disponivel\n", id);
    }
    pthread_mutex_unlock(&mutexPilhas);
  } else {
    printf("Inquilino %d tentou alugar imovel porem ja aluga um imovel\n",
id);
  }
}
```

A função *alugarImovel* é responsável por vincular o imóvel a ser alugado com o respectivo inquilino.

Caso ainda não tenha havido aluguel por parte do inquilino em questão, há a verificação se é o único imóvel disponível em sua pilha, ou não, para que os ponteiros sejam corretamente manipulados e por fim é impresso o *id* do inquilino e o *código* do imóvel alugado.

Caso já tenha feito um aluguel anterior e ainda não o tenha devolvido, é impresso que houve essa tentativa mas sem êxito - uma vez que houve a decisão do grupo de permitir apenas um aluguel por inquilino.

```
void entregarImovel(int id, Imovel **imovel){
 if((*imovel) != NULL) {
   char * codigo = (*imovel)->codigo;
   pthread_mutex_lock(&mutexPilhas);
   (*imovel)->anterior = topoEntregues;
   topoEntregues = (*imovel);
   if((*imovel)->anterior == NULL) {
     fundoEntregues = (*imovel);
   }
   pthread_mutex_unlock(&mutexPilhas);
   (*imovel) = NULL;
   printf("Inquilino %d entregou imovel %s\n", id, codigo);
 } else {
   printf("Inquilino %d tentou entregar imovel porem nao esta alugando
imovel no momento\n", id);
 }
}
```

A função *entregarImovel* é responsável por devolver um imóvel alugado por um inquilino, o qual em seguida poderá ser disponibilizado para futuros novos aluguéis.

Caso o inquilino de fato tenha um aluguel ocorrendo, há a verificação se é o único imóvel entregue em sua pilha, ou não, para que os ponteiros sejam corretamente manipulados e por fim é impresso o *id* do inquilino e o *código* do imóvel entregue.

Caso contrário, é impresso que houve essa tentativa mas sem êxito.

```
void disponibilizarImoveis(int id) {
 pthread_mutex_lock(&mutexPilhas);
 if (fundoEntregues != NULL) {
   fundoEntregues->anterior = topoDisponiveis;
   if(fundoEntregues == NULL) {
     fundoDisponiveis = fundoEntregues;
   }
   fundoEntregues = NULL;
   topoDisponiveis = topoEntregues;
```

```
    topoEntregues = NULL;
    printf("Corretor %d disponibilizou imoveis entregues\n", id);
  } else {
    printf("Corretor %d tentou disponibilizar imoveis porem nenhum foi
entregue\n", id);
  }
  pthread_mutex_unlock(&mutexPilhas);
}
```

A função *disponibilizarImoveis* é responsável por pegar os elementos da pilha de imóveis entregues e passar para a pilha de imóveis disponíveis, permitindo que agora sejam novamente alugados.

Há a verificação se há imóveis entregues em sua pilha, para que os ponteiros sejam corretamente manipulados e por fim é impresso o *id* do corretor que executou esta ação.

Caso contrário, é impresso que houve essa tentativa mas sem êxito.

```
void *acoesInquilino(void *arg)
{
  int id = *(int *)arg;
  struct Imovel *imovelPtr;
  while (1)
  {
    alugarImovel(id, &imovelPtr);
    sleep(5 + rand() % 4);
    entregarImovel(id, &imovelPtr);
  }
}
```

A função *acoesInquilino* é responsável por realizar a tentativa de aluguel de um imóvel e, posteriormente - após tempo randomizado de 5 a 9 segundos - a tentativa de entrega de um imóvel alugado.

```
void *acoesCorretor(void *arg)
{
  int id = *(int *)arg;
  while (1)
  {
    int i = rand() % 5;
    switch (i)
    {
    case 0:
```

```
      disponibilizarImoveis(id);
      break;
    case 1 || 2 || 3:
      criarImovel(id);
      break;
    case 4:
      removerImovel(id);
      break;
    }
    sleep(3 + rand() % 4);
  }
}
```

A função *acoesCorretor* é responsável por realizar a randomizar as tentativa de ações de um corretor, disponibilizando um imóvel, criando um imóvel ou removendo um imóvel. A randomização ocorre de 0 a 4, tendo a função de criar imóvel 3 vezes mais chance de ocorrer para que possibilite mais ações no sistema tanto de inquilino quanto de corretor.

```
void printarDisponiveis(){
  printf("-------------- printando imoveis disponiveis --------------\n");
  Imovel *temp = topoDisponiveis;
  while(temp != NULL) {
    printf("Codigo do imóvel disponivel é: %s\n", temp->codigo);
    temp = temp->anterior;
  }
}
```

A função *printarDisponiveis* é responsável por imprimir na tela o *código* dos imóveis que foram disponibilizados.

```
void printarEntregues() {
  printf("-------------- printando imoveis entregues --------------\n");
  Imovel *temp = topoEntregues;
  while(temp != NULL) {
    printf("Codigo do imóvel entregue é: %s\n", temp->codigo);
    temp = temp->anterior;
  }
}
```

A função *printarEntregues* é responsável por imprimir na tela o *código* dos imóveis que foram entregues.

```c
int main(int argc, char const *argv[])
{
 int qtdC = 50;
 int qtdI = 50;
 pthread_t corretor[qtdC], inquilino[qtdI];
 int cIds[qtdC];
 int iIds[qtdI];

 for (size_t i = 0; i < qtdC; i++)
 {
   cIds[i] = i;
   pthread_create(&corretor[i], NULL, acoesCorretor, &cIds[i]);
 }

 for (int i = 0; i < qtdI; i++)
 {
   iIds[i] = i;
   pthread_create(&inquilino[i], NULL, acoesInquilino, &iIds[i]);
 }

 pthread_exit(NULL);

 return 0;
}
```

A função main executa o sistema, e inicia atribuindo as quantidades de corretores e inquilinos, criando as threads de corretor para os corretores e as threads de inquilinos para os inquilinos, atribuindo os Ids únicos para corretores e inquilinos.

Casos de testes executados

Teste #1: 20 segundos

Composição:

- 2 inquilinos;
- 2 corretores.

Saída:

```
Inquilino 0 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 1 tentou alugar imovel porem nao ha imovel disponivel
CORRETOR 0 adicionou Imóvel #0
---------------------------------
 Codigo: #0
 Endereço: R. Maria Eduarda, 345
 Preço: RS 2040,00
 Bairro: Trindade
---------------------------------
Inquilino 0 tentou entregar imovel porem nao esta alugando imovel no momento
Inquilino 0 alugou imovel #0
Inquilino 1 tentou entregar imovel porem nao esta alugando imovel no momento
Inquilino 1 tentou alugar imovel porem nao ha imovel disponivel
CORRETOR 0 adicionou Imóvel #1
---------------------------------
 Codigo: #1
 Endereço: R. Conselheiro Mafra, 123
 Preço: RS 1999,99
 Bairro: Pantanal
---------------------------------
CORRETOR 1 adicionou Imóvel #2
---------------------------------
 Codigo: #2
 Endereço: R. João Vieira, 456
 Preço: RS 2500,00
 Bairro: Agronomica
---------------------------------
Corretor 0 tentou disponibilizar imoveis porem nenhum foi entregue
Inquilino 0 entregou imovel #0
Inquilino 0 alugou imovel #2
Inquilino 1 tentou entregar imovel porem nao esta alugando imovel no momento
Inquilino 1 alugou imovel #1
. . .
```

Teste #2: 2 segundos

Composição:

- 30 inquilinos;
- 30 corretores.

```
CORRETOR 3 adicionou Imóvel #0
```

```
---------------------------------
 Codigo: #0
 Endereço: R. João Marçal, 234
 Preço: RS 2500,00
 Bairro: Pantanal
---------------------------------
Corretor 5 tentou disponibilizar imoveis porem nenhum foi entregue
Corretor 12 tentou disponibilizar imoveis porem nenhum foi entregue
Corretor 7 tentou disponibilizar imoveis porem nenhum foi entregue
CORRETOR 8 adicionou Imóvel #1
---------------------------------
 Codigo: #1
 Endereço: R. Maria Eduarda, 345
 Preço: RS 1999,99
 Bairro: Trindade
---------------------------------
CORRETOR 10 removeu imovel #1
CORRETOR 16 adicionou Imóvel #2
---------------------------------
 Codigo: #2
 Endereço: R. Tangara, 567
 Preço: RS 2040,00
 Bairro: Capoeiras
---------------------------------
Corretor 20 tentou disponibilizar imoveis porem nenhum foi entregue
CORRETOR 24 adicionou Imóvel #4
---------------------------------
 Codigo: #4
 Endereço: R. João Marçal, 234
 Preço: RS 1250,00
 Bairro: Centro
---------------------------------
Corretor 21 tentou disponibilizar imoveis porem nenhum foi entregue
CORRETOR 23 adicionou Imóvel #3
---------------------------------
 Codigo: #3
 Endereço: R. João Marçal, 234
 Preço: RS 2760,30
 Bairro: Pantanal
---------------------------------
CORRETOR 25 adicionou Imóvel #5
---------------------------------
 Codigo: #5
 Endereço: R. Conselheiro Mafra, 123
 Preço: RS 1999,99
 Bairro: Agronomica
---------------------------------
CORRETOR 27 removeu imovel #5
Inquilino 13 alugou imovel #3
Inquilino 14 alugou imovel #4
Inquilino 15 alugou imovel #2
```

```
Inquilino 16 alugou imovel #0
Inquilino 17 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 18 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 19 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 20 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 21 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 22 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 23 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 24 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 25 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 26 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 27 tentou alugar imovel porem nao ha imovel disponivel
Corretor 28 tentou disponibilizar imoveis porem nenhum foi entregue
Inquilino 28 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 29 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 0 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 1 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 2 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 3 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 4 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 5 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 6 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 7 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 8 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 9 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 10 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 11 tentou alugar imovel porem nao ha imovel disponivel
Inquilino 12 tentou alugar imovel porem nao ha imovel disponivel

. . .
```

Teste #3: 40 segundos

Composição:

- 1 inquilino;
- 1 corretore.

```
Inquilino 0 tentou alugar imovel porem nao ha imovel disponivel
Corretor 0 tentou disponibilizar imoveis porem nenhum foi entregue
Inquilino 0 tentou entregar imovel porem nao esta alugando imovel no momento
Inquilino 0 tentou alugar imovel porem nao ha imovel disponivel
CORRETOR 0 adicionou Imóvel #0
---------------------------------
 Codigo: #0
 Endereço: R. Maria Eduarda, 345
 Preço: RS 2040,00
 Bairro: Trindade
---------------------------------
Inquilino 0 tentou entregar imovel porem nao esta alugando imovel no momento
Inquilino 0 alugou imovel #0
```

```
Inquilino 0 entregou imovel #0
Inquilino 0 tentou alugar imovel porem nao ha imovel disponivel
CORRETOR 0 adicionou Imóvel #1
---------------------------------
 Codigo: #1
 Endereço: R. Maria Eduarda, 345
 Preço: RS 1999,99
 Bairro: Trindade
---------------------------------
Inquilino 0 tentou entregar imovel porem nao esta alugando imovel no momento
Inquilino 0 alugou imovel #1
Corretor 0 disponibilizou imoveis entregues
Inquilino 0 entregou imovel #1
Inquilino 0 alugou imovel #0
Inquilino 0 entregou imovel #0
CORRETOR 0 tentou remover imovel porem nao ha imoveis disponiveis
Inquilino 0 tentou alugar imovel porem nao ha imovel disponivel
```