Sistemas Operativos

Trabalho Prático

2024/2025

Licenciatura de Engenharia Informática

Rúben Miguel Gomes Ferreira – 2021129743

1. Indice

2. Estruturas

Estruturas utilizadas:

Prompt - Responsável por manusear os comandos intruzidos no decorrer dos programas

```c
typedef struct {
    char comando[MAX_MSG];
    char arg1[MAX_MSG];
    char arg2[MAX_MSG];
    char arg3[MAX_MSG];
} prompt;
```

Mensagens - Responsável por manusear as informações de entidades no Jogo

```c
typedef struct {
    char remetente[MAX_NAME_LENGTH];
    char topico[MAX_NOME_TOPICO];
    int duracao;
    char mensagem[MAX_MSG];
} mensagem;
```

Users - Responsável por armazenar as informações do Labirinto

```c
typedef struct {
    char username[MAX_NAME_LENGTH];
    char fifo[300];
    char topics[MAX_TOPICS][MAX_NOME_TOPICO];
    int topics_count;
} users;
```

Topic - Responsável por manusear as informações partilhadas entre Jogo e Motor

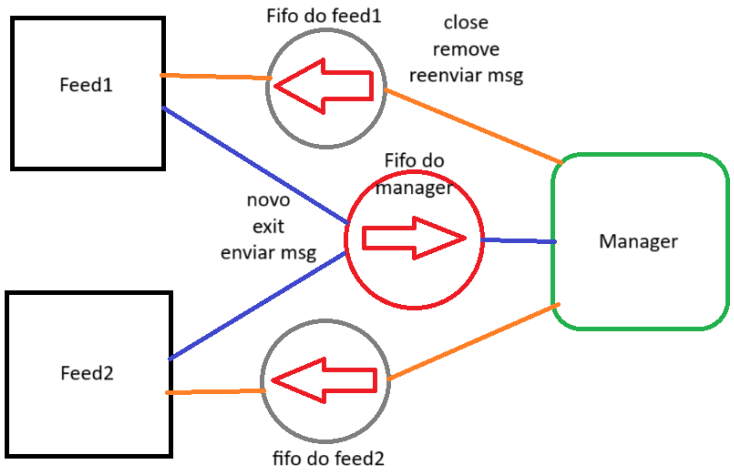
```c
typedef struct {
    char topic[MAX_NOME_TOPICO];
    int nPersistentes;
    int locked;
} topic;
```

3. Funcionamento

Desenvolvi o programa feed e manager de modo a existirem segundo o modelo **cliente-servidor**, onde:

- O `manager` atua como servidor, centralizando a gestão de tópicos e mensagens.
- Cada `feed` representa um cliente, interagindo com o `manager` para subscrever tópicos, publicar mensagens e receber atualizações.

Cada ação no feed tem consequências no manager, de modo a atualizar a informação global dos programas em execução. Esta atualização da informação é realizada através de fifos, seguindo o esquema abaixo:



A informação enviada do manager para um feed, é enviada para um fifo, sendo que cada feed tem o seu ficheiro fifo, onde recebe informação destinada apenas a ele.

Toda a informação enviada de um feed para o manager é enviada para um unico fifo identificando o feed que enviou, de modo ao manager conseguir alterar a informação relativamente ao cliente certo.

4. Armazenamento

O feed armazena localmente os topicos a que está subscrito e os topicos existentes no manager. Esta informação é comunicada ao manager e mantida atualizada por ele (no caso de algum tópico ser bloqueado, o número de mensagens persistentes ser alterado, ou deixar de existir.

```
char subedtopic[MAX_TOPICS][MAX_NOME_TOPICO]; //topicos subs pelo feed
int numsubTopics = 0;

topic topics[MAX_TOPICS]; //topicos existentes no manager
int numTopics = 0;
```

O manager armazena informações sobre todos os utilizadores, os topicos existentes, e as mensagens persistentes existentes. Ficando responsável por atualizar os feeds de qualquer alteração.

```
int numUtilizadores = 0;
users Users[MAX_USERS]; //feeds existentes

mensagem Persistentes[MAX_TOTAL_PERSIS]; //Mensagens Persistentes
int numPersistentes = 0;

topic topics[MAX_TOPICS]; //topicos existentes
int numTopics = 0;
```

5. Funções

De modo a gerir toda a informação, comunicação entre programas e input de comandos, desenvolvi as seguintes funções:

Manager:

```
    //FUNCOES MANAGER
void *admin_thread(); //input do usuario

void *feed_monitor_thread(void *arg); //comunicações dos feeds

void *decrease_msg_duration_thread(); //tempo das mensagens persistentes

void ResendMessage(mensagem msg, bool modopersistentes, const char *Vremetente); //Reenvio de mensagens

void handle_fifo_comum_manager(const char *message); // intrepertar comunicações dos feeds

void close_notify_feeds(); //notificar close

void ManageTopic(char *topic_nome, int locked_status, int nPersistentes); //atualizar topicos nos feeds

int has_persistent_messages(const char *topic); //verificar se existem mensagens permanentes

//Comandos Manager
void handle_users(); //comando users

void handle_remove(prompt *cmd); //comando remove

void handle_manager_topics(); //comando topics

void handle_show(const prompt *cmd); //comando show

void handle_lock(prompt *cmd); //comando lock

void handle_unlock(prompt *cmd); //comando unlock

void help_manager(); //comando help
```

Feed:

```c
    //FUNCOES FEED

void *input_thread(void *arg); // input do usuario

void *receive_thread(void *arg); // comunicações do manager

void notify_manager(const char *username, int status); //avisar join/exit

void handle_fifo_feed(char *message); // intrepertar comunicação manager


//Comandos Feed
void handle_feed_topics(); //comando topics

void handle_msg(prompt *cmd, const char *nome_jogador); //comando msg

void handle_subscribe(prompt *cmd, const char *nome_jogador); //comando subscribe

void handle_unsubscribe(prompt *cmd, const char *nome_jogador); //comando unsubscribe

void help_feed(); //comando help
```

A maioria das funções trabalha em conjunto de modo a alterar um estado/dado de determinada informação, e informar o manager/feed dessa alteração. Por exemplo, um feed quando subscreve um topico (utilizando o comando 'subscribe') atualiza o seu array de topicos subscritos, e informa o manager que subscreveu esse tópico. O manager, como mantem dados sobre os clientes, guarda esta informação, para, por exemplo, reencaminhar mensagens desse topico para quem o tem subscrito.