Licenciatura de Engenharia Informática

Departamento de Engenharia Informática e de Sistemas

Trabalho Prático de Sistemas Operativos

Meta 2

Realizado por:

Acácio Agabalayeve Coutinho – 2020141948

José Pedro Sousa Almeida – 2020141980

Conteúdo

Introdução:

O objetivo deste trabalho pratico é a implementação de uma plataforma de leilões. Neste relatório vamos clarificar as principais ações tomadas e alguns pormenores de implementação.

Funcionamento do programa:

O backend é ligado em primeiro lugar, pois sem ele não é possível existirem clientes/frontends. Não faz sentido existirem mais do que um backend em simultâneo pelo que apenas é possível ligar um backend. Depois de um backend ligado é possível ligarem-se até 20 frontends, desde que estes sejam de utilizadores diferentes. Logo após a execução do backend são lançados os promotores que periodicamente vão emitir promoções. Após isto os tanto os clientes/frontends como o administrador podem utilizar o sistema. Tanto os mecanismos de comunição como as threads aqui envolvidas são explicadas mais á frente.

Comunicação:

A comunicação entre o backend e frontend é feita através de named pipes. Quando é iniciado o backend, é criado o FIFO "SERVIDOR" que é um FIFO que vai servir para os frontends enviarem informação para o backend e para o backend ler essa mesma informação. Já quando um frontend é iniciado, é criado um FIFO "Cliente%d" em que o "%d" é o PID do frontend respetivo. Este FIFO vai servir para o backend enviar informação para o frontend pretendido e para esse mesmo frontend receber a informação.

O frontend envia para o backend uma estrutura do tipo "Comando" (*figura 2*) em que o "comando" é uma flag que indica qual operação fazer (licitar um item, adicionar dinheiro etc…).

O backend envia para o frontend uma estrutura do tipo "Resposta" (*figura 1*) em que o "comando" indica a resposta ao pedido do frontend.

```
typedef struct resposta
{
    int num;
    Item item;
    int comando;
    pid_t pid;
} Resposta;
```

Figura 1 - estrutura Resposta

```
typedef struct {
    Item item;
    User user;
    pid_t pid;
    int comando;
}Comando;
```

Figura 2- estrutura Comado

Threads e Select:

No frontend existe um thread que possibilita leitura e tratamento da informação por parte do backend enquanto o cliente pode escrever comandos.

No backend um mecanismo **Select**, que permite o tratamento dos pedidos que chegam através dos FIFOS dos clientes e permite a escrita de comandos. Existe também uma thread que é responsável pelo tempo do sistema. Existem ainda mais N threads que servem para tratar dos promotores existentes.

Para o bloqueio e proteção dos dados é utilizado o mutex.

Variávies de ambiente:

Antes de inciar o programa é obrigatório executar o "*script.sh*" para fazer o export das variáveis de ambiente.

```
export FPROMOTERS=promotores.txt
export FUSERS=utilizadores.txt
export FITEMS=items.txt
export HEARTBEAT=20
```

Figura 3 – Variáveis de ambiente

Opções de implementação:

- Os utilizadores autenticados ficam guardados numa lista de utilizadores de tamanho 20.

- Para implementar o mecanismo que deteta que um cliente terminou sem avisar a plataforma foi utilizada a função *alarm()* juntamente com o sinal *SIGALARM* para enviar periodicamente um sinal para o backend. Do lado do backend, foi criada uma lista com o pid e tempo restante desde o último sinal recebido desse utilizador. Esse tempo vai sendo decrementado até que chega um novo sinal e este é atualizado ou até este chegue a 0 e nesse caso o utilizador é removido dessa lista e da lista de utilizadores ativos.

- Tanto do lado do backend como do frontend estão implementados sinais que controlam a saída dos programas através do CRTL + C

- Sempre que o sistema é fechado corretamente (com o comando close), o tempo é guardado no ficheiro FITEMS para ser recuperado nas próximas execuções.

Makefile:

Este é o makefile que compila todos os programas necessários.

```
all: frontend backend

frontend: frontend.o utils.o
	gcc frontend.o utils.o -o frontend

backend: backend.o utils.o users_lib.o
	gcc backend.o utils.o users_lib.o -o backend

backend.o: backend.c backend.h utils.h users_lib.h
	gcc -c backend.c -o backend.o

frontend.o: frontend.c utils.h
	gcc -c frontend.c -o frontend.o

utils.o: utils.c utils.h
	gcc -c utils.c -o utils.o

limpa:
	echo "A limpar..."
	rm backend.o frontend.o utils.o
```

Conclusão:

Com a realização deste trabalho tivemos a oportunidade de trabalhar num ambiente Unix, o que nunca tinha acontecido antes e o que veio a ser uma agradável surpresa. Em suma foi bastante interessante a realização deste trabalho prático, que nos permitiu perceber melhor os conceitos nele abordados como por exemplo comunicação entre processos e programar com threads.