Instituto Superior de Engenharia de Coimbra
Licenciatura em Engenharia Informática
Sistemas Operativos

Trabalho Prático – Pacman
Programação em C para Unix

Diogo Santos nº 21210835

Índice

Introdução

O presente trabalho é desenvolvido no âmbito da unidade curricular Sistemas Operativos, a qual se encontra inserida no programa de estudos do 1º semestre, do 2º ano da Licenciatura em Engenharia Informática.

A realização deste trabalho tem como objetivo implementar um sistema multi jogador do jogo Pacman em tempo real.

A implementação é realizada em linguagem C, especificamente para os sistemas operativos Unix, sendo necessário utilizar alguns dos recursos que a API do Unix disponibiliza.

De referir que, os jogadores estarão todos na mesma máquina Unix, e por isso não serão utilizados mecanismos de comunicação em rede, mas sim mecanismos de comunicação inter-processo.

O jogo estará dividido em dois programas, sendo eles um cliente e um servidor. O servidor irá fazer toda a gestão do jogo, ao passo que o cliente apenas envia comandos ao servidor e apresenta as respostas ao jogador, não tendo qualquer influência nas ações de jogo.

As regras do jogo do Pacman são as normais e conhecidas por todos.

Estrutura e Implementação do Servidor

O servidor corresponde ao processo responsável por gerir todos os clientes, desde o decidir quem é *pacman* ou *glutão* até ao avaliar os movimentos de cada jogador como válidos ou não.

O programa começa com a preparação do jogo:

- Criação do mapa;
- Criação de *threads* - para possibilitar que os *glutões* se movam de forma autónoma;
- Escolha da entrada dos dados - seja esta feita por teclado no caso dos comandos do servidor, ou pelo *FIFO* no caso do servidor receber um pedido do cliente, para isto é utilizada a funcionalidade *select*.

A partir daqui o programa apenas avalia os pedidos recebidos.

Estruturas utilizadas

- A estrutura *PEDIDO* - é apenas usada no inicio do jogo para detetar se um determinado cliente já efetuou o login ou não, para guardar o seu nome, palavra passe e pid, atribuir uma posição de jogo e uma personagem (pacman ou glutão).

- A estrutura *RESPOSTA_MAPA* - apenas é usada para o servidor mandar a informação do mapa para o cliente, como por exemplo paredes, migalhas, estimulantes, etc.

- A estrutura REFRESH_POS - apenas serve para receber o pedido de atualização de coordenadas da parte do cliente para serem confirmadas do lado do servidor.

```
//PEDIDO
typedef struct {

  int login;
  char nome[30], passwd[30];
  int  pid, posX, posY;
  char player[3]; // glu / pac

} PEDIDO;

//RESPOSTA MAPA
typedef struct{

    int pacmanX, pacmanY;
    int pid;
    char mat[22][19];

} RESPOSTA_MAPA;

//RFRESH POS
typedef struct{

  int pid;
  int x;
  int y;

}REFRESH_POS;
```

Fig. 1

Comunicação

O servidor tem dois pipes, *mapa.bin* e *dados.bin*. O *mapa.bin* serve para receber pedidos depois do cliente estar logado, como por exemplo o refrescamento da sua posição. O *dados.bin* recebe pedidos apenas de autenticação.

O esquema abaixo apresentado é um exemplo de um jogo já em execução, ou seja, o "Cliente" já está logado e pronto a jogar enquanto que o "Novo Cliente" ainda está na fase de autenticação.

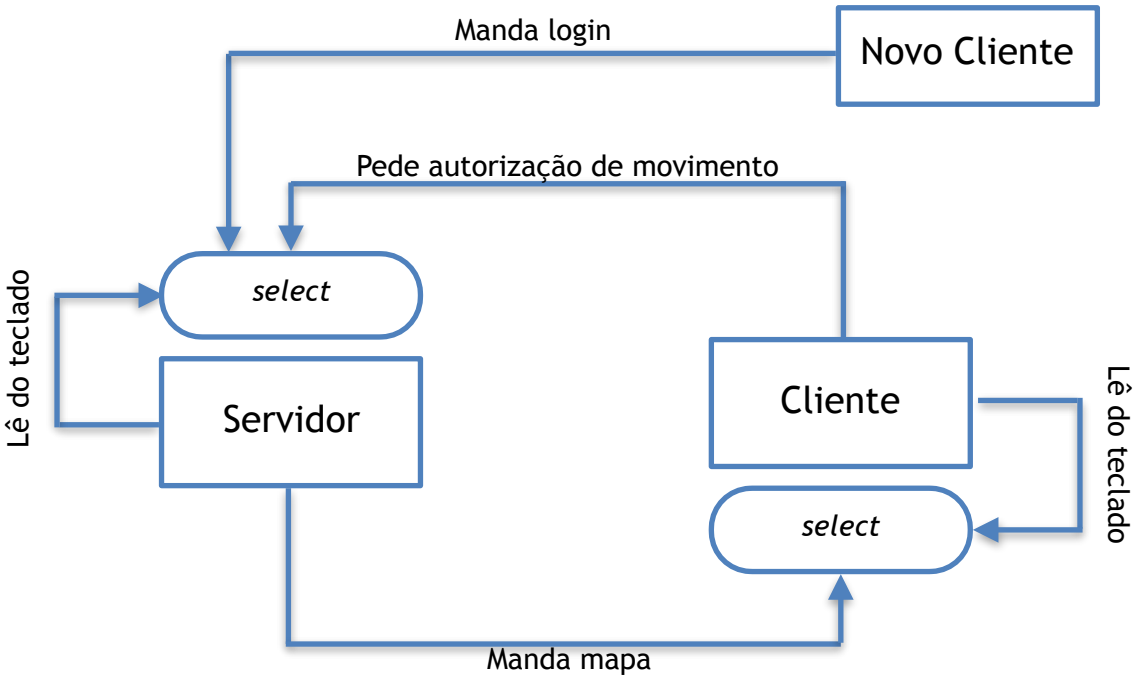

Fig. 2

O *select* do lado do servidor está a atender pedidos de três *file descriptors*, nomeadamente o teclado do lado do servidor, o pipe *mapa.bin* e o pipe *dados.bin*.

O *select* do lado do cliente por sua vez atende dois *file descriptors*, nomeadamente o teclado para movimentar o seu personagem e o seu pipe para receber as devidas atualizações do servidor tanto logins como atualizações do mapa.

Comandos

add - cria uma conta nova com o username e password especificados verificando a sua existência, caso seja verdade não será adicionada;

users - mostra a lista de utilizadores ativos no jogo e respetivos PIDs;

kick - termina a ligação ao jogador identificado, o utilizador não é apagado mas é removido do jogo;

shutdown - desliga o servidor apenas, não guarda informação.

Estrutura e Implementação do Cliente

O cliente apenas recebe os comandos introduzidos pelo utilizador, isto apenas se já existir um servidor em execução e a sua autenticação for aceite pelo mesmo.

Este utiliza um mecanismo semelhante ao do servidor, a funcionalidade *select*, como podémos ver ilustrado na *Fig.1,* pois estará a receber comandos do utilizador respetivo ao mesmo tempo que lhe é enviada informação do mapa pelo servidor, respetiva a outros jogadores ou *threads*. Além disso, é o cliente que interpreta várias funções da biblioteca *ncurses* que são traduzidas em tempo real para o servidor tais como o pressionar das teclas de direção.

Outras funcionalidades implementadas

Foi também implementada a funcionalidade de encerrar tanto o servidor como o cliente usando o tratamento de sinais. Os programas reconhecem o *Ctrl + C* para terminar os programas da devida forma, isto é, fechar os pipes e terminar as funcionalidades de memória de vídeo, no caso do cliente, da biblioteca *ncurses*.

Conclusão

A realização do trabalho prático proposto constituiu uma mais-valia no meu percurso de aprendizagem da API para UNIX, tendo permitido consolidar os conteúdos lecionados na unidade curricular de Programação e um "maior à-vontade" na escolha de arquitetura de um determindo problema.

Face à realização deste trabalho, penso que o objetivo inicialmente traçado - implementar um sistema multi jogador do jogo Pacman em tempo real - foi marioritariamente alcançado.

As maiores dificuldades encontradas durante a execução foram:

- Utilização da funcionalidade *select* com múltiplos *file descriptors* que me levou algum tempo a dominar;
- Leitura e escrita de diferentes tipos de dados no mesmo *file descriptor*;
- Dominio de *threads*;
- Mobilização de conhecimentos da teoria para a prática;
- Gestão de tempo.

Assim o desenvolvimento deste trabalho foi muito enriquecedor para a minha formação enquanto aluno de Sistemas Operativos.