Sistemas Operativos

2017/2018

Trabalho Prático - Meta 3

Hugo Moreira – 21240034

Filipe Ribeiro – 21220620

Índice

Estruturas de Dados

```
17
18 #define CHAR 30
19 #define TAM 8
20 char str[1024];
21
22 typedef struct labirinto labi;
23 typedef struct  PLAYER play;
24 struct labirinto
25 {
26   int largura, altura;
27  // char bloco[100];//DESTRUTIVEL
28  // char parede[100] ;//INDESTRUTIVEL
29   char lab[MAXX][MAXY];
30 };
31
32 struct PLAYER
33 {
34  char nome[30];
35  char passw[30];
36  int check;
37  int pid;
38  int ch;
39 };
40
41 typedef struct{
42         int num, x, y;
43         labi a;
44
45 }RESPOSTA;
46
47 // Guarda Clientes
```

O nosso trabalho terá cinco estruturas:

Labirinto guarda as informações do labirinto, com os blocos destrutíveis e o mapa de carateres a apresentar no ecrã;

Player guarda a informação do jogador e irá server na comunicação com o servidor, guardando o nome do Jogador, a password (para o login), uma variável para a confirmação do login, o PID do jogador, e a tecla selecionada pelo jogador;

Resposta guarda as estruturas Posição com as posições dos personagens, o código do personagem a controlar 'num', o numero de bombas que possui e de super bombas, e uma estrutura Labirinto com o labirinto atualizado;

Posicao guarda a posição x, y, que usamos para a posição das personagens;

Jogador guarda a informação dos personagens, contem a estrutura Labirinto, para as validações nas Threads, um numero único 'num' (tipo Id), 'humano' para verificar se é controlado por um Cliente, 'tempo' para ter um tempo de espera entre cada movimento na Thread, um Char 'cara' para o caracter a mostrar na consola, 'fim' para saber se o personagem terminou o Jogo;

```
/ Guarda Clientes
play clientes[TAM];
```

Todos os Clientes ligados ao Servidor serão guardados num Array de estruturas PLAYER clientes;

Arquiteturas e Estratégia de Named Pipes

```
261         int fd, fd_resp, ctl = 0, ret, i;
262         char cmd[20];
263
264
265         fd_set rfds;
266         struct timeval tv;
267
268         //Garantir que só existe um servidor
269         if(access(FIFO_SERV, F_OK)==0){
270                 printf("Ja existe um servidor a correr...\n");
271                 exit(1);
272         }
273
274         signal(SIGINT, sinal_shutdown); // ctrl + c
275
276         mkfifo(FIFO_SERV, 0600);
277
278         fd = open(FIFO_SERV, O_RDWR);
279
```

Iremos confirmar que já não existe nenhum Servidor a correr e termina caso já exista, se não existir cria um FIFO e abre para receber pedidos;

```
                                }
                        }else printf("COMANDO ERRADO\n" );
                }else
                        printf("COMANDO ERRADO\n" );
        }

        if(FD_ISSET(fd, &rfds))
        {
          //Ler dados do FIFO...
                i=read(fd, &p, sizeof(p));
                if(i==sizeof(p))
                {
                        if(p.check == 0){
                                inilogin(p,&res);
                        }
                        else{
                                // Apresenta Dados no Servidor
                                printf("Chegou um pedido [%d] bytes)\n",i);
                                //printf("\n%s %s \n;" , p.nome, p.passw);
                                //Guarda Cliente no Servidor
                                        int existe = 0;
                        int livre = -1;
                        for(i=0; i<TAM; i++)
                        {
                        if(clientes[i].pid == 0 && livre == -1)
                                livre = i;
                        if(clientes[i].pid == p.pid)
                                {existe = 1;
                                clientes[i].pid =p.pid ;
                                strcpy(clientes[i].nome,p.nome);
```

O Servidor recebe através do FIFO uma estrutura PLAYER um pedido, caso a variável 'check' seja 0 este inicia o a função "inilogin" que irá confirmar os dados do login, se for 1 passa para a atualização do mapa e das posições dos Jogadores;

```
                        strcpy(clientes[livre].nome,p.nome);
                        cli++;
                }

                        actualizadados(&res,a, p);
                        }

                sprintf(str, "ccc%d", p.pid);
                fd_resp=open(str, O_WRONLY);

        // Envio da Resposta ao Cliente
                write(fd_resp, &res, sizeof(res));
                //printf(" %d ", getpid());
                //printf("%d",res.num);
                close(fd_resp);

        }

        }
}
```

Após receber o pedido e realizar o processamento dos dados, este abre e envia através de um FIFO criado no Cliente, a estrutura RESPOSTA e envia para o Cliente através do seu PID de maneira a que apenas aquele cliente o receba, e fecha o FIFO do Cliente;

```
70      if(access(FIFO_SERV, F_OK)!=0)
71      {
72              printf("O servidor nao esta a correr...\n");
73              return 1;
74      }
75
76      p.pid = getpid();
77      printf("%d\n",p.pid);
78      sprintf(str, "ccc%d", p.pid);
79      mkfifo(str, 0600);
80
81      //abrir fifo do servidor
82      fd = open(FIFO_SERV, O_WRONLY);
83
84      signal(SIGUSR1, sinal);
85      signal(SIGINT, sinal_shutdown); // ctrl + c
86
87      //---------------Login-----------------------------------------------------------------
88      do{
89
90              inicialogin(&p);
91              //printf("\ncheck = %d\n", p.check);
92              printf("\n%s %s \n" , p.nome, p.passw);
93              //enviar pedido ao servidor...
94              write(fd, &p, sizeof(p));
95              // Recebe Resposta...
96              fd_resp=open(str, O_RDONLY);
97              read(fd_resp, &res, sizeof(res));
98              p.check = res.num;
99
100             close(fd_resp);
101             }while(p.check == 0);
```

O Cliente quando Inicia verifica se existe algum Servidor ativo, caso não exista termina, cria um FIFO com o seu PID de maneira a que só ele receba as respostas e não todos os Clientes, abre o FIFO do Servidor para enviar os pedidos;

No ciclo do Login inicia a função “inicialogin”, que pede o nome e a password, e envia para o Servidor a estrutura PLAYER com as suas informações, abre e recebe uma estrutura RESPOSTA pelo FIFO que criou, atualiza o ‘check’, caso seja ‘1’ as credenciais estão corretas, fecha o FIFO;

```
22
23          // Grava a tecla
24          p.ch = getchar();
25
26          //enviar pedido ao servidor...
27          write(fd, &p, sizeof(p));
28
29          // Recebe Resposta...
30          fd_resp=open(str, O_RDONLY);
31          read(fd_resp, &res, sizeof(res));
32
33          imprime(res.a);
34          close(fd_resp);
35
36
37          attron(COLOR_PAIR(2));
38          mvaddch(res.y, res.x, 'A');
39          refresh();
40
41          }while(1);
42          // fechar fifo do servidor
43          close(fd);
44          unlink(str);
45          pthread_join(le,NULL);
46          endwin();
47          return 0;
48 }
49
```

Dentro do ciclo principal, no qual o utilizador já se encontra a jogar, o Cliente envia a estrutura PLAYER para o Servidor e recebe uma estrutura RESPOSTA com as informações atualizadas e imprime o mapa;

Funcionalidades Realizadas

```
        initscr();

        noecho();
        cbreak();
        keypad(stdscr, TRUE);
        curs_set(0);

        start_color();
        init_pair(1, COLOR_BLACK, COLOR_GREEN);
        init_pair(2, COLOR_RED, COLOR_GREEN);
        init_pair(3, COLOR_BLUE, COLOR_BLACK);


        do{
        attron(COLOR_PAIR(3));
        mvaddch(res.y, res.x, ' ');

        attron(COLOR_PAIR(1));
        // Grava a tecla
        p.ch = getchar();

        //enviar pedido ao servidor...
        write(fd, &p, sizeof(p));

        // Recebe Resposta...
        fd_resp=open(str, O_RDONLY);
        read(fd_resp, &res, sizeof(res));

        imprime(res.a);
        close(fd_resp);


        attron(COLOR_PAIR(2));
        mvaddch(res.y, res.x, 'A');
        refresh();

        }while(1);
        // fechar fifo do servidor
        close(fd);
        unlink(str);
        pthread_join(le,NULL);
        endwin();
        return 0;
}
```

Após ser confirmado o login do Jogador será iniciado as ncurses, não irá ser apresentado no ecrã a tecla dada pelo utilizador, não será preciso carregar no enter para validar, e não irá aparecer o cursor;

Irá iniciar as cores definidas para o labirinto ficar a verde e preto e o carater 'A' controlado pelo utilizador a vermelho;

```c
fd_set rfds;
struct timeval tv;

//Garantir que só existe um servidor
if(access(FIFO_SERV, F_OK)==0){
        printf("Ja existe um servidor a correr...\n");
        exit(1);
}

signal(SIGINT, sinal_shutdown); // ctrl + c

mkfifo(FIFO_SERV, 0600);

fd = open(FIFO_SERV, O_RDWR);

do{     setbuf(stdout,NULL);
        FD_ZERO(&rfds);
        FD_SET(0, &rfds); // toma atenção ao teclado
        FD_SET(fd, &rfds); // toma atenção ao fifo
        tv.tv_sec = 1; // tempo de espera!
        tv.tv_usec = 5000;

        ret = select(fd+1, &rfds, NULL, NULL, &tv); // espera
        if(ret == 0)
        {
                //printf(".");
                fflush(stdout);
        }else
        if(ret > 0)
        {
                if (FD_ISSET(0, &rfds))
                {
                        //Ler dados do teclado
                        scanf( " %[^\n]s",comando);
                if ( (palavra = contapalavras(comando) ) == 2)
                {
                        sscanf(comando,"%s",comandoo);

                        if(strcmp(comandoo,"add")==0)
                        {
```

Temos um Select no Servidor para este Receber os Pedidos dos Clientes e processar e enviar as Respostas e também receber os comandos no próprio Servidor;

Este também envia Respostas automáticas para todos os Clientes ligados ao Servidor, e o servidor contem também um Select para receber as Respostas do cliente e apresentar no ecrã e para receber do teclado.

```c
void sinal_shutdown(int sinal){
        int i;

        printf("Recebi o sinal %d (crtl+c), vou desligar...\n", sinal);

        for(i=0;i<TAM;i++){
            if(clientes[i].pid!=0)
                kill(clientes[i].pid, SIGUSR1);
        }
        unlink(FIFO_SERV);
        sleep(2);
        exit(1);
}
```

Caso o utilizador faça um control + c no Servidor este processa o sinal apresentando uma mensagem e enviando um sinal SIGUSR1 a todos os Clientes ligados ao Servidor;

```c
void sinal(int sinal){
        clear();
        endwin();
        printf("Foste expulso da sessao\n");
        unlink(str);
        sleep(1);
        exit(1);
}

void sinal_shutdown(int sinal){

        clear();
        endwin();
        printf("A sair...\n");
        unlink(str);
        sleep(1);
        exit(1);
```

Caso o Cliente receba um sinal SIGUSR1 este encerra as ncurses e desliga o FIFO criado e apresenta uma mensagem no ecrã;

Threads usadas

Temos 4 Threads, uma para cada personagem, com a função mover() que irá controlar o seu movimento quando não é controlado por nenhum Cliente.

Comportamentos Anómalos

Encontramo-nos com anomalias de código na parte do login do cliente em que se introduzirmos o nome e password com informação o correta na parte do nome ele valida bem sem perguntar pela password.

Quando um cliente sai da sessão o Servidor termina porque este irá enviar a Resposta automática para um Cliente que já não existe;

As bombas não estão implementadas para rebentarem;