1 Servidor GS - Organização de dados

O servidor GS instalado na máquina 'tejo' do laboratório LT5, no porto 58011, usa uma estrutura de directorias para armazenar toda a informação referente ao protocolo de forma persistente entre sessões.
Os alunos podem replicar total ou parcialmente a estrutura de directorias aqui descrita, a qual foi concebida tendo em vista a simplificação do processamento para armazenamento flexível e indexação de jogos e *scores*, aproveitando ainda as estruturas de dados previstas no *filesystem* do Linux para obter facilmente a ordenação de ficheiros nele contidos.
O presente guia exemplifica também, em linguagem C, a obtenção de conteúdos de directorias ordenados e conversões de tempos e datas úteis para a realização do projecto.

1.1 A directoria de trabalho do servidor GS

Na directoria de trabalho do servidor GS encontram-se duas directorias designadas GAMES e SCORES. Assume-se que no arranque do servidor GS estas duas directorias já estão criadas.

1.1.1 A directoria GAMES

A directoria GAMES contém um ficheiro por cada jogo em curso contendo o estado do jogo, e ainda as directorias de jogadores, contendo cada uma delas os resumos de todos os jogos já terminados para um dado jogador.
O estado de um jogo em curso para um dado PLID encontra-se guardado num ficheiro designado GAME_(plid).txt em que (plid) é sub-

stituído pelo *player ID* em formato fixo de 6 dígitos.

Os jogos já terminados pertencentes a um dado PLID ficam guardados, para cada player, numa directoria cuja designação é (plid) sob a directoria GAMES. O caminho para a localização dos ficheiros de jogos finalizados é assim: GAMES/(plid).

O ficheiro GAME_(plid).txt com o estado do jogo em curso tem o seguinte formato:

Campos da primeira linha:

PPPPPP M CCCC T YYYY-MM-DD HH:MM:SS s

em que

PPPPPP - PLID com seis dígitos em formato fixo

M - Modo. Uma letra em formato fixo. Pode ser P (Play) ou D (Debug)

CCCC - Código de cores a decifrar - 4 letras em formato fixo.

T - Tempo em segundos para terminar em formato variável. De 1 a 3 dígitos terá valor máximo 600

YYYY-MM-DD - É a data de início do jogo

HH:MM:SS - É a hora de início do jogo

s - Momento de início do jogo em segundos fornecido pela função **time**().

Em cada uma das linhas subsequentes, o ficheiro de jogo contém uma jogada com o seguinte formato:

T: CCCC B W s

Onde

T: significa trial e CCCC é a sequência de cores da jogada. B é um dígito indicando o número de cores na posição certa e W é o número de cores fora da posição certa.

O campo s indica o momento em segundos, desde o início do jogo em que a jogada foi realizada.

Quando o jogo é encerrado, o ficheiro acima descrito é movido para a directoria do jogador que o criou (GAMES/(plid)/), sendo-lhe acrescentada no final (após as linhas referentes às jogadas) uma última linha com o seguinte formato:

YYYY-MM-DD HH:MM:SS s

Esta última linha contém a data e hora de encerramento do jogo. E ainda um campo (s) com duração efectiva do jogo em segundos. Ao ser transferido para a directoria GAMES/(plid), o ficheiro GAME_(plid).txt assume uma nova designação com o formato:

YYYYMMDD_HHMMSS_(code).txt

Em que YYYYMMDD e HHMMSS é a data/hora de criação deste ficheiro, e code é uma letra de quatro possíveis: W, F, Q ou T, e que indica como foi finalizado o jogo: por Win, Fail, Quit ou Timeout respectivamente.

Exemplo:
O player com o PLID=123456 executou o comando *start* 123456 300. Ao receber a mensagem SNG correspondente ao comando *start*, o servidor criou o ficheiro GAME_123456.txt com primeira linha:

123456 P YYBG 300 2024-11-14 22:41:10 1731624070

Esta linha significa que o servidor gerou aleatoriamente o código de cores YYBG para ser decifrado. O jogo terá uma duração máxima de 300 segundos (por escolha do utilizador) com data/hora de início: 2024-11-14 22:41:10, ou 1731624070 segundos após um momento de referência considerado pela função **time**().
Depois o jogador emitiu o comando:
try R G B Y
que teve como resultado a sequência 1 2. Esta jogada foi registada aos 35s após o início do jogo. A linha gravada no ficheiro de jogo foi assim:
T: RGBY 1 2 35
As linhas seguintes registadas no ficheiro foram:
T: GGPO 0 1 47

T: YYBG 4 0 58

Com esta última jogada o jogo foi terminado com sucesso. Então, o servidor transferiu o ficheiro de jogo para a directoria GAMES/123456, gravando no ficheiro uma linha adicional:

2024-11-14 22:42:08 58

Indicando a data de fecho do jogo e o número de segundos que o mesmo durou.
Ao ficheiro acima descrito, guardado na directoria GAMES/123456 foi dada a designação 20241114_224208_W.txt indicando a data e a hora de finalização do jogo e que o mesmo foi terminado com sucesso (codigo W).

Esta metodologia de designação dos jogos já terminados vai permitir ao servidor ordenar facilmente os jogos já terminados por ordem decrescente de data de finalização. Assim o jogo mais recente vai figurar no topo da lista.

1.1.2 A directoria SCORES

Quando um dado jogo termina com sucesso, é também criado um ficheiro de 'score' na directoria SCORES. Este ficheiro de 'score' é único para cada jogo terminado com sucesso.
A designação do ficheiro de 'score' é:
score_PLID_DDMMYYYY_HHMMSS.txt

em que score poderá ter um valor entre 001 e 100 (por exemplo) sempre com três dígitos e PLID é a identificação do jogador. DDMMYYYY e HHMMSS são a data e a hora a que o jogo terminou com sucesso respectivamente.

A designação acima facilita a obtenção da lista de 'scores' ordenados por valor decrescente de 'score'.

O ficheiro scores tem uma única linha com o seguinte formato:

SSS PPPPPP CCCC N mode

Onde SSS é o score obtido com três dígitos (formato fixo), PPPPPP é o PLID com 6 dígitos, CCCC é o código de cores a decifrar, N é o número de jogadas que ocorreram até à jogada de sucesso e mode pode assumir uma de duas designações: PLAY ou DEBUG consoante o jogo tiver sido iniciado com start ou debug, respectivamente.
Para o caso do jogo exemplificado, o ficheiro de scores gerado ficou com a designação:

062_123456_14112024_224208.txt

Contendo uma única linha:

062 123456 YYBG 3 PLAY

1.2 Ordenação de ficheiros em directorias por nome

Listas ordenadas de ficheiros de jogos finalizados dentro de cada directoria de PLID podem ser obtidas recorrendo à função **scandir**().

A função FindLastGame() que se lista abaixo, é usada pelo GS em execução no tejo para obter o ficheiro de jogo mais recente para um dado PLID. O nome do ficheiro é retornado para a variável apontada por fname.

```
int FindLastGame(char *PLID, char *fname)
{
    struct dirent **filelist;
    int n_entries,found;
    char dirname[20];

    sprintf(dirname,"GAMES/%s/",PLID);

    n_entries = scandir(dirname, &filelist , 0, alphasort);

    found=0;

    if (n_entries <= 0)
        return(0);
    else
    {
        while (n_entries--)
        {
            if(filelist[n_entries]->d_name[0]!='.' && !found)
            {
                sprintf(fname,"GAMES/%s/%s",PLID,filelist[n_entries]->d_name);
                found=1;
            }
            free(filelist[n_entries]);
        }
        free(filelist);
    }

    return(found);
```

}

A função FindTopScores() listada abaixo é usada pelo GS em execução no tejo para ter acesso a cada um dos 10 ficheiros com scores mais elevados contidos na directoria SCORES.

```
int FindTopScores(SCORELIST *list)
{
    struct dirent **filelist;
    int n_entries, i_file;
    char fname[300];
    FILE *fp;
    char mode[8];

    n_entries = scandir("SCORES/", &filelist, 0, alphasort);

    if (n_entries <= 0)
        return(0);
    else
    {
        i_file=0;
        while (n_entries--)
        {
            if(filelist[n_entries]->d_name[0]!='.' && i_file<10)
            {
                sprintf(fname,"SCORES/%s",filelist[n_entries]->d_name);
                fp=fopen(fname,"r");
                if(fp!=NULL)
                {
                    fscanf(fp,"%d %s %s %d %s",
```

```
                    &list->score[i_file],list->PLID[i_file],list->col_code[i_file],&list->no_tries[i_file],mode);

                    if(!strcmp(mode,"PLAY"))
                        list->mode[i_file]=MODE_PLAY;
                    if(!strcmp(mode,"DEBUG"))
                        list->mode[i_file]=MODE_DEBUG;

                    fclose(fp);
                    ++i_file;
                }
            }
            free(filelist[n_entries]);
        }
        free(filelist);
    }

    list->n_scores=i_file;
    return(i_file);
}
```

A definição das estruturas relativas a directorias nas funções acima ilustradas estão contidas no ficheiro 'dirent.h'.

1.3 Envio de ficheiros solicitados pelos comandos *show_trials* e *scoreboard*

Os ficheiros de texto que o GS envia para o *player* na sequência da recepção de uma mensagem STR ou SSB já se encontram formatados para apresentação ao utilizador. O *player* recebe o ficheiro em causa por TCP, grava-o localmente, e de seguida imprime o ficheiro no terminal linha a linha, efectuando uma leitura não formatada do ficheiro (p. ex. com **fgets**()). Todas as linhas do ficheiro vêm terminadas com o caracter '\n' não significando o mesmo, neste contexto, o fim da mensagem.

1.4 Aspectos de fiabilização do protocolo UDP

O protocolo da camada de transporte UDP usado no presente projecto não é fiável, não garantindo a entrega das mensagens, nem a preservação da sequência pela qual elas foram enviadas. Por isso se dotou o protocolo da funcionalidade de numeração das mensagens TRY e das respectivas respostas RTR.

Quando o servidor recebe uma mensagem TRY com o número de jogada esperado (seja ele N), ele responde a essa mensagem TRY com uma mensagem RTR contendo o número de jogadas acumuladas N. E ao incrementar o contador de jogadas acumuladas, o servidor fica à espera da jogada N+1. Simultaneamente, a aplicação player recebendo essa confirmação da jogada N, incrementa o número de próxima jogada para N+1.

O servidor GS instalado no tejo, responderá com uma mensagem RTR INV\n sempre que receber uma mensagem TRY com um número de *trial* que não seja o esperado. Com uma única excepção: Se essa mensagem TRY, trouxer o número da última jogada recebida e se a sequência de cores que ela transporta for igual à sequência de cores recebida na última jogada registada. Esta excepção destina-se a contemplar a situação em que o jogador, não tendo recebido a confirmação da última jogada registada volta a repetir a referida jogada. No caso de se verificar esta excepção, o servidor responde à aplicação player repetindo a mesma resposta RTR OK que presumivelmente estará em falta na aplicação player.

Concomitantemente, a aplicação player deve ignorar confirmações RTR com número de jogada que já tenha recebido.

Quando o servidor recebe uma mensagem TRY, com o número de jogada por si esperado mas contendo uma sequência de cores igual à de outra jogada anterior, o servidor responde à aplicação player com RTR DUP\n indicando que já recebeu uma jogada com o mesmo conteúdo e não incrementando o contador de jogadas acumuladas. Neste caso, a aplicação player também não deve incrementar o contador de jogadas, pois a mensagem RTR DUP\n indica que o servidor ignorou a jogada recebida.

Quando o servidor recebe uma mensagem SNG (ou DBG) para início de um novo jogo, e já existe um jogo iniciado mas sem jogadas registadas, o servidor assume que a confirmação de início de jogo que enviou anteriormente ao cliente se perdeu e que este decidiu emitir

um novo pedido de início. Nesse caso, o servidor aceita a nova mensagem SNG (ou DBG) iniciando um novo jogo com os parâmetros que constarem dessa nova mensagem.

1.5 Temporização de jogos e seu controlo no servidor GS

O servidor GS instalado no tejo controla a temporização dos jogos de uma forma muito simples:
Em princípio, qualquer jogo iniciado ficará activo até que um evento de acesso ao jogo no servidor verifique que o jogo já está expirado.

Assim, em qualquer acção de acesso a um jogo, decorrente da recepção de uma mensagem (SNG, DBG, TRY, QUT ou STR), a primeira verificação do servidor será sobre a temporização do jogo.
Se o tempo de validade do jogo tiver expirado e o jogo ainda não estiver dado como encerrado, o servidor vai em primeiro lugar encerrar o jogo. Logo de seguida, o servidor vai então executar a operação requerida considerando o estado do jogo resultante da operação de verificação de temporização que efectuou prioritariamente.
Os alunos podem utilizar os procedimentos aqui descritos nos seus servidores para verificar a temporização dos jogos. Para tal pode ser útil a informação na secção que se segue sobre o procesamento de datas:

1.6 Processamento de tempos e datas

A execução da função **time**(&fulltime), retorna na variável designada fulltime que é do tipo **time_t**, o número de segundos decorridos desde a data 1970-01-01 00:00:00 até ao momento presente. Recomenda-se cuidado na utilização do tipo **time_t**, nomeadamente na sua interação com inteiros. Nalgumas situações, o tipo **time_t** pode interagir com inteiros sem precauções especiais. Os alunos devem verificar caso a caso e nas máquinas que vão usar para o desenvolvimento as precauções a tomar para o efeito. O número de segundos decorridos desde 1970-01-01 00:00:00 até ao momento em que o presente projecto vai ser avaliado não excede a capacidade de um inteiro de 32 bits (4 bytes) com sinal.

A conversão do número de segundos decorridos desde 1970-01-01 00:00:00 até um dado momento para o formato YYYY-MM-DD HH:MM:SS consegue-se usando a função **gmtime**() como se exemplifica a seguir:

```
#include<time.h>

time_t fulltime;
struct tm *current_time;
char time_str[20];


time(&fulltime); // Get current time in seconds starting at 1970- ...
current_time = gmtime(&fulltime); // Convert time to YYYY-MM-DD HH:MM:SS. current_time points to a struct of type tm
sprintf(time_str,"%4d-%02d-%02d %02d:%02d:%02d,
                                current_time->tm_year+1900,current_time->tm_mon+1,current_time->tm_mday,
                                current_time->tm_hour,current_time->tm_min,current_time->tm_sec);
```