1 Servidor ES - Organização de dados

O servidor ES instalado na máquina 'tejo' do laboratório LT5 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 eventos e reservas, 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 ES

Na directoria de trabalho do servidor ES encontram-se duas directorias designadas USERS e EVENTS. Assume-se que no arranque do servidor ES estas duas directorias já estão criadas.

Ilustra-se a seguir a estrutura da base de dados do servidor, em que ESDIR é a directoria onde se encontra o executável ES.

```
ESDIR
├── USERS
│   ├── (uid1)
│   │   ├── (uid1)_pass.txt
│   │   ├── (uid1)_login.txt
│   │   ├── CREATED
│   │   │   ├── (eid1).txt
│   │   │   └── (eid2).txt
│   │   └── RESERVED
│   │       ├── R-(uid1)-(date1).txt
│   │       ├── R-(uid1)-(date2).txt
│   │       ├── R-(uid1)-(date3).txt
│   │       └──
│   ├── (uid2)
│   ├── (uid3)
│   └──
└── EVENTS
    ├── (eid1)
    │   ├── START_(eid1).txt
    │   ├── RES_(eid1).txt
    │   ├── DESCRIPTION
    │   │   └── (desc_fname1)
    │   ├── END_(eid1).txt
    │   └── RESERVATIONS
    │       ├── R-(uid1)-(date1).txt
    │       ├── R-(uid2)-(date2).txt
    │       ├── R-(uid3)-(date3).txt
    │       └──
    ├── (eid2)
    ├── (eid3)
    └──
```

O servidor ES criará dentro da directoria USERS uma directoria por cada utilizador que se regista. A designação da directoria de utilizador coincide com o UID do utilizador em causa. Do mesmo modo, o servidor ES criará dentro da directoria EVENTS uma directoria por cada evento criado. A designação da directoria de evento coincide com o EID do evento em causa.

- Na directoria de cada utilizador (sob USERS) são criados:

- Um ficheiro (uid)_pass.txt que contém a password do utilizador. Este ficheiro existirá enquanto o utilizador permanecer registado.
- Um ficheiro (uid)_login.txt indicando que o utilizador está em sessão. Este ficheiro existe apenas enquanto o utilizador estiver em sessão.
- Uma directoria designada CREATED contendo informação sobre todos os eventos criados pelo utilizador. A cada evento iniciado pelo utilizador corresponde um ficheiro dentro da directoria CREATED.
- Uma directoria designada RESERVED contendo informação sobre todos os eventos nos quais o utilizador reservou lugares. A cada evento no qual o utilizador efectuou uma reserva, corresponde um ficheiro dentro da directoria RESERVED. Um duplicado deste ficheiro existe também sob a directoria RESERVATIONS de cada evento.

Quando um utilizador remove o seu registo (comando **unregister**), os ficheiros (uid)_pass.txt e (uid)_login.txt e apenas estes são removidos, preservando-se as respectivas directorias e o seu conteúdo.
Se o utilizador voltar a registar-se, será criado um novo ficheiro (uid)_pass.txt, assumindo-se que no novo registo, o utilizador herdará toda a informação guardada nas directorias CREATED e RESERVED em momentos anteriores. O efeito prático resultante de um utilizador cancelar o seu registo e voltar a registar-se é equivalente à alteração da password.
Os ficheiros guardados nas directorias CREATED e RESERVED destinam-se a permitir ao servidor recuperar de forma rápida a lista de todos os eventos relevantes para cada utilizador evitando buscas exaustivas sobre a directoria EVENTS, contendo esta última informação mais completa embora organizada de outra forma.

- Na directoria de cada evento (sob EVENTS) é criada uma directoria por cada evento criado. Cada directoria de evento conterá:

- Um ficheiro START_(eid).txt que contém toda a informação relevante sobre o evento. Este ficheiro é criado no momento de criação do evento quando o utilizador emite o comando **create**.
- Um ficheiro RES_(eid).txt que contém o número total de reservas acumuladas para o evento. Este ficheiro é criado com valor 0 quando o utilizador emite o comando **create**.
- Um directoria DESCRIPTION que contém o ficheiro desc_fname o qual contém uma imagem ou uma descrição do evento criado. Este ficheiro é carregado por TCP para o servidor quando o utilizador emite o comando **create**.
- Um ficheiro END_(eid).txt que só é criado quando o evento é dado como encerrado.
- Uma directoria RESERVATIONS que contém todas as reservas efectuadas para o evento em causa. A directoria RESERVATIONS conterá um ficheiro por cada operação de reserva de lugares. O nome de cada ficheiro de reserva contém o UID do utilizador que fez a reserva e a data de recepção da reserva no servidor para permitir a recuperação de todas as reservas ordenadas por data. Os ficheiros contidos na directoria RESERVATIONS são cópias dos ficheiros contidos nas directorias RESERVED dos *users* que efectuaram reservas no evento em causa.

1.2 Formatos dos ficheiros que contêm informação relevante

Descreve-se aqui o conteúdo dos ficheiros da base de dados. Os ficheiros _login.txt e aqueles que estão na directoria CREATED, valem por apenas estarem presentes, independentemente da informação que possam conter.
Descreve-se a seguir a organização da informação relevante contida nos restantes ficheiros:

O ficheiro _pass.txt contém a password do utilizador. Se o utilizador emitir o comando **unregister**, o ficheiro _pass.txt é apagado.

O ficheiro START_ contém uma única linha com os seguintes campos:

UID event_name desc_fname event_attend start_date start_time

em que:
UID é a identificação do utilizador com formato fixo de seis caracteres numéricos.
name é o nome dado pelo utilizador ao evento em causa.
desc_fname é o nome do ficheiro de descrição do evento.
event_attend é o número total de lugares do evento.
start_date é a data de início do evento no formato dd-mm-yyyy HH:MM

Se existir um ficheiro END_, ele indica que o evento se encontra encerrado.
Nesse caso, o ficheiro END_ contém uma linha única com os seguintes campos:

end_datetime

em que:
end_datetime é a data de encerramento do evento no formato dd-mm-yyyy HH:MM:SS

Os ficheiros contidos na directoria RESERVED têm a mesma designação e conteúdo dos ficheiros que se encontram sob a directoria RESERVATIONS. O nome destes ficheiros é:
R-(uid)-(date)_(time).txt
Em que (uid) é o UID do utilizador que efectuou a reserva, (date) é a data em que foi efectuada a reserva no formato YYYY-MM-DD e (time) é a hora da reserva no formato HHMMSS. Se por absurdo um utilizador humano conseguir emitir duas reservas com um intervalo inferior a um segundo, o segundo ficheiro gerado irá sobrepôr-se ao primeiro se ambos tiverem a mesma designação. No entanto isso não constitui problema pois os ficheiros sob as directorias RESERVED e RESERVATIONS têm valor meramente informativo. O ficheiro RES_(eid).txt contém o real totalizador das reservas para um evento.
O conteudo de cada ficheiro de reserva é:

UID res_num res_datetime

em que:
UID é a identificação do utilizador que efectuou a reserva.
res_num é o número de lugares reservados.
res_datetime é a data da reserva em causa no formato DD-MM-YYYY HH:MM:SS.

O estado de um evento não se encontra gravado em ficheiro algum. Sempre que necessário, o servidor ES calcula o estado do evento a partir dos factores que o determinam.

O ficheiro END_.txt que assinala a terminação de um evento é criado sempre que o utilizador que criou o evento decide fechá-lo ou sempre que o servidor, em acesso à base de dados percebe que a data do evento se encontra no passado. No primeiro caso de terminação, a data de fecho do evento é a data real na qual o evento foi encerrado. No segundo caso, a data de fecho do evento é a sua data de ocorrência.

1.3 Exemplo de execução

Para clarificar a organização de dados apresenta-se a seguir um exemplo de interacção entre aplicações clientes *user* e o servidor *ES*.

- O utilizador 111111 começou por registar-se emitindo o comando login pela primeira vez. Deu então início a um evento com a designação 'Conference' carregando o ficheiro 'Conference_01.jpg' e estipulando para este evento uma data de ocorrência 05-12-2025 15:00. O servidor ES notificou a aplicação user do sucesso da operação e atribuiu ao evento acabado de criar a identificação 001.
- Posteriormente, numa outra aplicação *user*, o utilizador 222222 registou-se emitindo o comando login pela primeira vez, e reservou 100 lugares no evento com EID==001.

A base de dados do servidor ES ficou com o seguinte conteúdo:

```
ESDIR
├── USERS
│   ├── 111111
│   │   ├── 111111_pass.txt
│   │   ├── 111111_login.txt
│   │   ├── CREATED
│   │   │   └── 001.txt
│   │   └── RESERVED
│   └── 222222
│       ├── 222222_pass.txt
│       ├── 222222_login.txt
│       ├── CREATED
│       └── RESERVED
│           └── R-222222-20251120_121505.txt
└── EVENTS
    └── 001
        ├── START_001.txt
        ├── RES_001.txt
        ├── DESCRIPTION
        │   └── Conference_01.jpg
        └── RESERVATIONS
            └── R-222222-20251120_121505.txt
```

Após esta interacção, o ficheiro RES_001.txt contém o valor 100.

A dada altura, o utilizador 222222 abandonou a sessão e o utilizador 333333 registou-se, tendo logo de seguida reservado 50 lugares no evento 001. Após a emissão dessa reserva, o utilizador 111111 decidiu encerrar o evento antecipadamente. Então o conteúdo da base de dados passou a ser:

```
ESDIR
├── USERS
│   ├── 111111
│   │   ├── 111111_pass.txt
│   │   ├── 111111_login.txt
│   │   ├── CREATED
│   │   │   └── 001.txt
│   │   └── RESERVED
│   ├── 222222
│   │   ├── 222222_pass.txt
│   │   ├── CREATED
│   │   └── RESERVED
│   │       └── R-222222-20251120_121505.txt
│   └── 333333
│       ├── 333333_pass.txt
│       ├── 333333_login.txt
│       ├── CREATED
│       └── RESERVED
│           └── R-333333-20251120_121510.txt
└── EVENTS
    └── 001
        ├── START_001.txt
        ├── RES_001.txt
        ├── DESCRIPTION
        │   └── Conference_01.jpg
        ├── END_001.txt
        └── RESERVATIONS
            ├── R-222222-20251120_121505.txt
            └── R-333333-20251120_121510.txt
```

No final destas interacções, o ficheiro RES_001.txt contém o valor 150.

2 Complementos de programação

Nesta secção sugerem-se procedimentos e exemplifica-se a utilização de algumas funções em C com a finalidade de facilitar a implementação do projecto.

2.1 Sobre o encerramento dos eventos

O servidor ES em operação no 'tejo' não toma a decisão de encerrar os eventos a determinados momentos. Não usa portanto temporizadores ou varrimentos periódicos à base de dados para encerrar os eventos.
Em vez disso, ele aproveita as interacções geradas pelas aplicações clientes que lhe acedem de forma concorrente, para verificar se um dado evento deve ser encerrado. Quando o servidor solicitado para uma dada operação que acede a um evento, verifica que já passou a data de encerramento para esse evento, cria na respectiva directoria o ficheiro END_. Se o utilizador que criou o evento, decide encerrá-lo, então o servidor ES, no cumprimento dessa ordem de encerramento antecipado, encerra o evento criando o ficheiro END_.
Se por exemplo, um utilizador pede o detalhe sobre um dado evento, o servidor aproveita essa solicitação (mensagem SED por TCP) para verificar se o evento ultrapassou o prazo de validade e em caso afirmativo, encerra o evento criando o ficheiro END_ mas enviando sempre o detalhe sobre o evento ao utilizador que o solicitou. Neste exemplo, o utilizador que solicitou o detalhe não se apercebe que o servidor ES aproveitou o seu pedido referente ao evento em causa para verificar se o prazo do mesmo tinha expirado.
Se um utilizador emitiu uma operação de RID para um dado evento, o servidor ES verifica em primeiro lugar se o evento já foi marcado como encerrado. Caso contrário, verifica de seguida se o evento já ultrapassou o seu prazo e responde em conformidade. Caso isso aconteça, marca o evento como encerrado. Caso o evento esteja dentro do prazo, cria os ficheiros respectivos nas directorias RESERVATIONS e RESERVED e responde afirmativamente ao utilizador.

Os alunos podem usar o procedimento acima descrito para controlar o estado dos eventos, ou podem usar outro que entendam conveniente como seja por exemplo a programação de temporizadores no ES orientada para esse fim.

2.2 Criação de directorias e subdirectorias

Exemplo de criação de directoria e subdirectorias para novo evento:

```
int CreateEVENTDir(int EID)
{
    char EID_dirname[15];
    char RESERV_dirname[25];
    char DESC_dirname[25];
    int ret;

    if(EID<1 || EID>999)
        return(0);

    sprintf(EID_dirname,"EVENTS/%03d",EID);

    ret=mkdir(EID_dirname,0700);
    if(ret==-1)
        return(0);



    sprintf(RESERV_dirname,"EVENTS/%03d/RESERVATIONS",EID);

    ret=mkdir(RESERV_dirname,0700);
    if(ret==-1)
    {
        rmdir(EID_dirname);
        return(0);
    }
```

```
    sprintf(DESC_dirname,"EVENTS/%03d/DESCRIPTION",EID);
    ret=mkdir(DESC_dirname,0700);
    if(ret==-1)
    {
        rmdir(EID_dirname);
        rmdir(RESERV_dirname);
        return(0);
    }
    return(1);
}
```

2.3 Criação e eliminação de ficheiro

Exemplos de funções de criação e eliminação do ficheiro de login:

```
int CreateLogin(char *UID)
{
    char login_name[35];
    FILE *fp;

    if(strlen(UID)!=6)
        return(0);

    sprintf(login_name,"USERS/%s/%s_login.txt",UID,UID);
    fp=fopen(login_name,"w");
    if(fp==NULL)
        return(0);
    fprintf(fp,"Logged in\n");
    fclose(fp);
    return(1);
```

```
}

int EraseLogin(char *UID)
{
    char login_name[35];

    if(strlen(UID)!=6)
        return(0);

    sprintf(login_name,"USERS/%s/%s_login.txt",UID,UID);
    unlink(login_name);
    return(1);
}
```

2.4 Obtenção de informações sobre ficheiro

Exemplo com função para determinação de existência e tamanho do ficheiro de descrição de evento

```
#include<stat.h>

int CheckDescFile(char *fname)
{
    struct stat filestat;
    int ret_stat;

    ret_stat=stat(fname, &filestat);

    if(ret_stat==-1 || filestat.st_size==0)
        return(0);
```

```
    return(filestat.st_size);
}
```

2.5 Listagem ordenada de conteúdos de directorias

Há situações que requerem a ordenação de informação contida em directorias. Para tal, o servidor ES no 'tejo' aproveita as funcionalidades do *filesystem* do Linux no que respeita à obtenção de listagens de directorias com os conteúdos ordenados. Exemplifica-se a seguir uma função hipotética GetEventList(). As entradas relevantes da directoria são carregadas uma a uma para a variável apontada por list.

```
#include <dirent.h>

int GetEventList(char *EID, EVENTLIST *list)
{
    struct dirent **filelist;
    int n_entries,no_events,i_ent=0;
    char dirname[55];

    sprintf(dirname,"USERS/%s/CREATED/",EID);
    n_entries = scandir(dirname, &filelist, 0, alphasort);

    no_events=0;

    if (n_entries <= 0)
        return(0);
    else
    {
        while (i_ent<n_entries)
        {
            if(filelist[i_ent]->d_name[0]!='.')
```

```
            {
                memcpy(list->event_no[no_events],filelist[i_ent]->d_name,3);
                list->event_no[no_events][3]=0;
                ++no_events;
            }
            free(filelist[i_ent]);
            ++i_ent;
        }
        list->no_events=no_events;
        free(filelist);
    }
    return(no_events);
}
```

Variantes da função GetEventList() podem ser usadas para seleccionar apenas a maior reserva, ou o evento

2.6 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 - também conhecido por *Unix time stamp*. 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 DD-MM-YYYY HH:MM:SS (data de calendário) 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 DD-MM-YYYY HH:MM:SS. current_time points to a struct of type tm
sprintf(time_str,"%2d-%02d-%04d %02d:%02d:%02d,
                                    current_time->tm_mday,current_time->tm_mon+1,current_time->tm_year+1900,
                                    current_time->tm_hour,current_time->tm_min,current_time->tm_sec);
```

Para comparação de datas pode ser útil converter uma determinada data de calendário na *Unix time stamp*. Para tal preenche-se uma estrutura do tipo **struct tm** com os valores relevantes e usa-se a função mktime(). Como se exemplifica a seguir para a conversão da data de calendário de um evento para *Unix time stamp*:

```
#include<time.h>

time_t creation_unixtime;
struct tm creation_datetime;

creation_datetime.tm_mday=event_day;
creation_datetime.tm_mon=event_month-1;
creation_datetime.tm_year=event_year-1900;
creation_datetime.tm_hour=event_hour;
creation_datetime.tm_min=event_min;
creation_datetime.tm_sec=0;

creation_unixtime=mktime(&creation_datetime);
```

2.7 Extensões ao protocolo

A secção do enunciado do projecto - **7 Open Issues** - sugere a possibilidade de extender o protocolo especificado. A directoria RESERVATIONS sob cada evento, acima descrita, não tem interesse prático na actual especificação do protocolo. Contudo, a informação nela contida permite recuperar rapidamente o detalhe das reservas efectuadas para cada evento.

Os alunos podem extender o protocolo na vertente da comunicação TCP para recuperar essa informação, adicionando comandos que permitam obtê-la com critérios de filtragem especificados nesses novos comandos. Como por exemplo: Todas as reservas efectuadas para um dado evento entre duas datas. Ou todas as reservas

efectuadas para um dado evento por um utilizador.