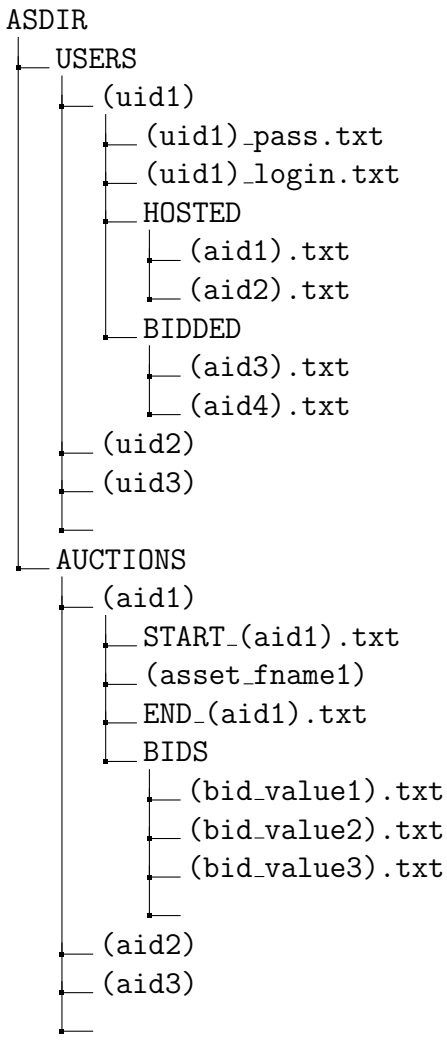
1 Servidor AS - Organização de dados

O servidor AS 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 leilões e licitações, 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 AS

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

Ilustra-se a seguir a estrutura da base de dados do servidor.


ASDIR
USERS
(uid1)
(uid1)_pass.txt
(uid1)_login.txt
HOSTED
(aid1).txt
(aid2).txt
BIDDED
(aid3).txt
(aid4).txt
(uid2)
(uid3)
AUCTIONS
(aid1)
START_(aid1).txt
(asset_fname1)
END_(aid1).txt
BIDS
(bid_value1).txt
(bid_value2).txt
(bid_value3).txt
(aid2)
(aid3)

O servidor AS 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 AS criará dentro da directoria AUCTIONS uma directoria por cada leilão criado. A designação da directoria de leilão coincide com o AID do leilão 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 durante a sessão do utilizador.
- Uma directoria designada HOSTED contendo informação sobre todos os leilões iniciados pelo utilizador. A cada leilão iniciado pelo utilizador corresponde um ficheiro dentro da directoria HOSTED.
- Uma directoria designada BIDDED contendo informação sobre todos os leilões nos quais o utilizador licitou. A cada leilão no qual o utilizador licitou, corresponde um ficheiro dentro da directoria BIDDED.

Quando um utilizador remove o seu registo, os ficheiros (uid)_pass.txt e (uid)_login.txt e apenas estes são removidos. 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 HOSTED e BIDDED em anteriores registos. O efeito prático resultante de um utilizador cancelar o seu registo e voltar a registar-se é apenas a alteração da password.
Os ficheiros guardados nas directorias HOSTED e em BIDDED não precisam de conter informação alguma em especial. Eles destinam-se apenas a permitir ao servidor recuperar de forma rápida a lista de todos os leilões relevantes para cada utilizador evitando buscas exaustivas sobre a directoria AUCTIONS, sendo que esta última contém informação mais completa embora esteja organizada de outra forma.

- Na directoria de cada leilão (sob AUCTIONS) é criada uma directoria por cada leilão iniciado. Cada directoria de leilão conterá:

- Um ficheiro START_(aid).txt que contém toda a informação relevante para o decurso do leilão. Este ficheiro é criado no momento de criação do leilão quando o utilizador emite o comando **open**.

- Um ficheiro asset_fname que contém uma imagem ou uma descrição do activo a ser leiloado. Este ficheiro é carregado por TCP para o servidor quando o utilizador emite o comando **open**.
- Um ficheiro END_(aid).txt que só é criado quando o leilão é dado como encerrado.
- Uma directoria BIDS que contém todas as licitações efectuadas para o leilão em causa. A directoria BIDS conterá um ficheiro por cada operação de licitação. O nome do ficheiro coincide com o valor licitado com formato fixo de 6 dígitos para permitir a recuperação de todas as licitações ordenadas por valor.

1.2 Formatos dos ficheiros que contêm informação relevante

Os ficheiros _pass.txt START_, END_ e aqueles contidos dentro da directoria BIDS de cada leilão contêm informação relevante para o servidor. Os ficheiros _login.txt e aqueles contidos nas directorias HOSTED e BIDDED valem por apenas existirem, independentemente da informação que possam conter. Descreve-se a seguir a organização da informação relevante contida nos ficheiros acima referidos:

O ficheiro _pass.txt contém a password do utilizador.

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

UID name asset_fname start_value timeactive start_datetime start_fulltime

em que:
UID é a identificação do utilizador com formato fixo de seis caracteres numéricos.
name é o nome dado pelo utilizador ao leilão em causa.
asset_fname é o nome do ficheiro de descrição do activo leiloado.
start_value é o valor base de licitação.
timeactive é a duração estipulada para o leilão em segundos.

start_datetime é a data de início do leilão no formato YYYY-MM-DD HH:MM:SS
start_fulltime é a data de início do leilão em segundos contados a partir de 1970-01-01 00:00:00 tal como obtido pela chamada da função time().

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

end_datetime end_sec_time

em que:
end_datetime é a data de encerramento do leilão no formato YYYY-MM-DD HH:MM:SS
end_sec_time é o tempo em segundos durante o qual o leilão esteve activo.

Os ficheiros contidos na directoria BIDS têm a designação VVVVVV.txt em que VVVVVV tem o formato fixo de 6 dígitos representando o valor do lance efectuado. Assim se torna fácil obter listagens do conteúdo da directoria BIDS ordenadas por ordem decrescente dos valores de licitação. Cada ficheiro com designação VVVVVV.txt possui uma única linha com os seguintes campos:

UID bid_value bid_datetime bid_sec_time

em que:

UID é a identificação do utilizador que efectuou o lance.
bid_value é o valor do lance em causa.
bid_datetime é a data da licitação em causa no formato YYYY-MM-DD HH:MM:SS.
bid_sec_time é o número de segundos decorridos desde o início do leilão até ao momento do lance em causa.

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 *AS*.

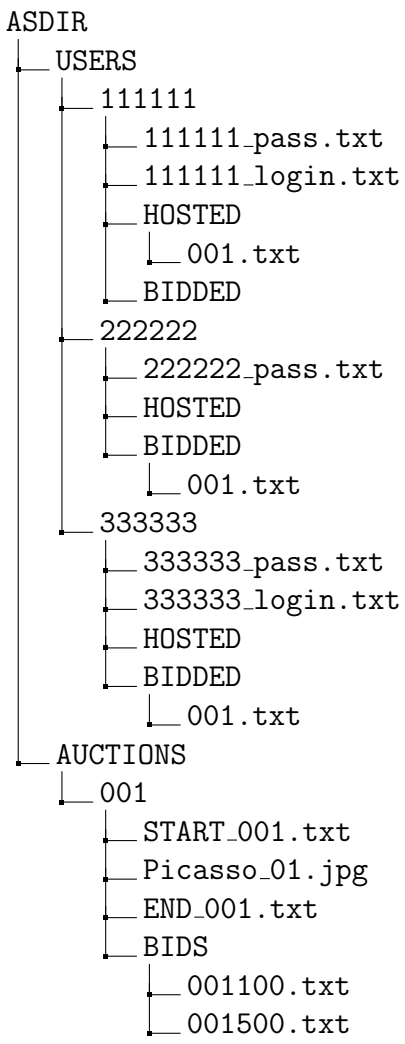
- O utilizador 111111 começou por registar-se emitindo o comando login pela primeira vez. Deu então início a um leilão com a designação 'Picasso' carregando o ficheiro 'Picasso_01.jpg' e estipulando para este leilão uma base de licitação 1000 e um tempo de duração do leilão de 3600 segundos. O servidor AS notificou a aplicação user do sucesso da operação e atribuiu ao leilão acabado de criar a identificação 001.
- Posteriormente, numa outra aplicação *user*, o utilizador 222222 registou-se emitindo o comando login pela primeira vez, e licitou no leilão com AID==001 com o valor 1100.

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

```
ASDIR
├── USERS
│   ├── 111111
│   │   ├── 111111_pass.txt
│   │   ├── 111111_login.txt
│   │   ├── HOSTED
│   │   │   └── 001.txt
│   │   └── BIDDED
│   └── 222222
│       ├── 222222_pass.txt
│       ├── 222222_login.txt
│       ├── HOSTED
│       └── BIDDED
│           └── 001.txt
└── AUCTIONS
    └── 001
        ├── START_001.txt
        ├── Picasso_01.jpg
        └── BIDS
            └── 001100.txt
```

A dada altura, o utilizador 222222 abandonou a sessão e o utilizador 333333 registou-se, tendo logo de seguida licitado no leilão 001 com um valor 1500. Após essa licitação, o utilizador 111111 decidiu encerrar o leilão antecipadamente. Então o conteúdo da base de dados

passou a ser:


ASDIR
USERS
111111
111111_pass.txt
111111_login.txt
HOSTED
001.txt
BIDDED
222222
222222_pass.txt
HOSTED
BIDDED
001.txt
333333
333333_pass.txt
333333_login.txt
HOSTED
BIDDED
001.txt
AUCTIONS
001
START_001.txt
Picasso_01.jpg
END_001.txt
BIDS
001100.txt
001500.txt

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 leilões

O servidor AS em operação no 'tejo' não toma a iniciativa de encerrar os leilões no prazo estipulado pelos utilizadores que os criaram. Não usa portanto temporizadores ou varrimentos periódicos à base de dados para verificar o prazo de validade dos leilões.
Em vez disso, ele aproveita os eventos gerados pelas aplicações clientes que lhe acedem de forma concorrente, para verificar se um dado leilão se encontra expirado. Quando o servidor verifica que um dado leilão se encontra expirado, cria na respectiva directoria o ficheiro END_. Se o utilizador que abriu o leilão, decide encerrá-lo, então o servidor AS, no cumprimento dessa ordem de encerramento antecipado, encerra o leilão criando o ficheiro END_.
Se por exemplo, um utilizador pede para descarregar o ficheiro de activo de um dado leilão, o servidor aproveita o evento criado pela chegada da mensagem SAS (por TCP) para verificar se o leilão ultrapassou o prazo de validade e em caso afirmativo, encerra o leilão criando o ficheiro END_ mas enviando sempre o ficheiro de activo ao utilizador que o solicitou. Neste exemplo, o utilizador que solicitou o ficheiro de activo não recebe qualquer notificação sobre o encerramento do leilão nem se apercebe que o servidor AS aproveitou o evento 'pedido de ficheiro de activo' referente ao leilão em causa para verificar se o prazo do mesmo tinha expirado.
Se por exemplo, um utilizador pediu uma qualquer listagem de leilões, o AS, após obter a listagem dos leilões verifica um a um se os leilões já foram marcados como encerrados verificando para cada um deles so o ficheiro END_ existe. Verifica também, para aqueles que ainda não tenham sido marcados como encerrados, se estão dentro do prazo de validade. E nesse processo, marca como encerrados aqueles que tenham ultrapassado o prazo. E na resposta ao pedido de listagem informa o utilizador do estado de cada leilão - se está activo ou encerrado.
Se um utilizador emitiu uma operação de BID sobre um dado leilão, o servidor AS verifica em primeiro lugar se o leilão já foi marcado como encerrado. Caso contrário, verifica de seguida se o leilão já ultrapassou o seu prazo. Caso isso aconteça, marca o leilão como encerrado e informa o utilizador de que o lance é recusado. Caso o leilão esteja dentro do prazo, cria os ficheiros respectivos nas directorias BIDS e BIDDED e responde afirmativamente ao utilizador.

Os alunos podem usar o procedimento acima descrito para controlarem o estado dos leilões, ou podem usar outro que entendam conveniente como seja por exemplo a programação de temporizadores no AS orientada para esse efeito.

2.2 Criação de directorias e subdirectorias

Exemplo de criação de directoria e subdirectoria para novo leilão:

```
int CreateAUCTIONDir(int AID)
{
    char AID_dirname[15];
    char BIDS_dirname[20];
    int ret;

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

    sprintf(AID_dirname,"AUCTIONS/%03d",AID);

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

    sprintf(BIDS_dirname,"AUCTIONS/%03d/BIDS",AID);

    ret=mkdir(BIDS_dirname,0700);
    if(ret==-1)
    {
        rmdir(AID_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 activo em leilão

```
#include<stat.h>

int CheckAssetFile(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 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);
```

2.6 Listagem ordenada de conteúdos de directorias

Há situações que requerem a ordenação de informação contida nas estruturas acima descritas. Nomeadamente,

- Obtenção do número mais alto de leilão iniciado.

- Obtenção do lance mais elevado para um dado leilão.

- Obtenção de uma lista com os 50 lances mais recentes para um dado leilão.

Para os casos acima enumerados, e por uma questão de simplificação, o servidor AS 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 a função GetBidList() que é usada pelo AS no 'tejo' para carregar todos os lances para o leilão AID por ordem decrescente do nome (valor) do lance. As entradas relevantes da directoria são carregadas uma a uma para a variável apontada por list.

```
#include <dirent.h>

int GetBidList(int AID, BIDLIST *list)
{
    struct dirent **filelist;
    int n_entries, n_bids, len;
    char dirname[20];
    char pathname[32];

    sprintf(dirname,"AUCTIONS/%03d/BIDS/",AID);
    n_entries = scandir(dirname, &filelist, 0, alphasort);
    if (n_entries <= 0)// Could test for -1 since n_entries count always with . and ..
        return(0);

    n_bids=0;
    list->no_bids=0;
    while (n_entries--)
    {
        len=strlen(filelist[n_entries]->d_name);
        if(len==10)// Discard '.', '..' and invalid filenames by size
        {
            sprintf(pathname,"AUCTIONS/%03d/BIDS/%s",AID,filelist[n_entries]->d_name);
            if(LoadBid(pathname,list))
```

```
                ++n_bids;
        }

        free(filelist[n_entries]);
        if(n_bids==50)
            break;
    }
    free(filelist);
    return(n_bids);
}
```

Variantes da função GetBidList() podem ser usadas para seleccionar apenas o lance mais elevado, ou o leilão mais recente existente na base de dados para atribuição de AID na abertura de um novo leilão. Ou para recolher das directorias BIDDED e HOSTED a relação dos leilões relevantes para um dado utilizador.