ISCTE IUL Instituto Universitário de Lisboa Escola de Tecnologias e Arquitetura Dep. de Ciências e Tecnologias de Informação	Sistemas Operativos Ano letivo 2019/2020 1º Semestre versão 3

Projeto Spotif-IUL (Parte 3)

Este enunciado detalha apenas as funcionalidades que devem ser implementadas na parte 3, correspondente à última matéria em avaliação na Unidade Curricular de Sistemas Operativos. Pretende-se avaliar a comunicação entre processos através de filas de mensagens, semáforos e memórias partilhadas.

A plataforma Spotif-IUL destina-se à partilha de músicas entre estudantes, docentes e colaboradores do ISCTE-IUL. Este programa (fictício!) pretende a estimular o estudo e o exercício físico. Para além de poderem ouvir as músicas preferidas, poderão também criar e partilhar playlists com os outros ouvintes.

Ouvinte: é a pessoa que deseja ouvir música e criar playlists. Tem um determinado perfil que inclui um *nickname, password, número, nome, e-mail, curso e playlists* a que está associado.

Música: é uma música que o ouvinte pode escolher, caracterizada pelo seguinte conjunto de atributos: *identificador da música* (da forma *MUS_número*), *nome da música*, *artista*, *duração* (em segundos), *ano de produção*, *género*, *posição no top*.

Playlist: conterá a lista de playlists criadas pelos ouvintes, caracterizada pelo seguinte conjunto de atributos: *identificador da playlist* (da forma *PL_número*), *nome da playlist*, *músicas* (identificador de músicas da forma *MUS_número*).

Os ouvintes ligam-se à aplicação Spotif-IUL, fazendo login com o seu *nickname* e a sua *password*, e escolhendo operações como listar músicas ou ouvir uma playlist. Um programa servidor irá tratar destes pedidos, enviando as respostas indicadas. O trabalho inclui também o desenvolvimento de uma aplicação admin que estará encarregue da manutenção das estruturas de dados e ficheiros envolvidos na parte do servidor.

Constituição dos grupos

O trabalho será realizado em grupos de 2 ou 3 estudantes (só em casos excecionais podem ser aceites grupos de um estudante). Durante o semestre, alunos e professores poderão reorganizar os grupos por forma a evitar grupos de 1 estudante. Os grupos devem começar por se registar usando o e-learning:

- e-learning da UC Sistemas Operativos
- Escolher Menu Grupos
- Escolher um grupo disponível **que tenha o nome do seu docente das aulas práticas**.

A constituição dos grupos deverá manter-se até ao final do semestre, mas caso se verifiquem alterações, deve ser criado um novo grupo, mediante prévia autorização dos docentes, que passará a ser usado em vez do anterior para submeter os trabalhos.

Entrega, relatório e autoavaliação

A entrega da Parte 3 do trabalho será realizada através da criação de **um** ficheiro ZIP (**ATENÇÃO:** não serão aceites ficheiros RAR, 7Z ou outro formato) onde estarão todos os ficheiros criados **INCLUINDO** o relatório. O relatório será entregue no formato TXT ou PDF. Não serão aceites documentos DOCX ou outro formato. O ficheiro ZIP em questão terá OBRIGATORIAMENTE o nome do grupo em minúsculas, e.g., "so-cc-g01-parte3.zip" ou "so-ac-g23-parte3.zip". Não serão aceites ficheiros com outros nomes que não o do grupo. Relembra-se que as iniciais do docente devem ser substituídas por aquelas a quem atribuiu o seu grupo:

- Prof. Carlos Coutinho – **cc**
- Prof. Ana Catarina Cruz – **ac**
- Prof. João Felício – **jf**
- Prof. Mário Rivotti – **mr**
- Prof. Paulo Pereira – **pp**

A entrega da Parte 3 do trabalho deverá ser feito por via eletrónica, através também do e-learning:

- e-learning da UC Sistemas Operativos
- Selecionam o link do grupo onde estão inscritos
- Selecionem o link "Trabalho Prático 2019/2020 Parte 3"
- Dentro do form "Visualizar Exercício de carregamento: Trabalho Prático 2019/2020 Parte 3", selecionem "Anexar Arquivo" e anexem o vosso ficheiro .zip. Podem submeter o vosso trabalho as vezes que desejarem, apenas a última submissão será contabilizada.

Observação. A submissão de um ficheiro substitui a versão anteriormente submetida. Realça-se que o formulário acima destina-se apenas à entrega da Parte 3.
Deverá ser criado um relatório em formato TXT ou PDF com informações sobre a elaboração do projeto. O relatório será entregue no formato TXT ou PDF. Não serão aceites documentos DOCX ou outro formato. Esse ficheiro deve chamar-se relatorio.txt ou relatorio.pdf, deverá ter aproximadamente 1 página A4 e deverá ser submetido dentro do ficheiro ZIP do trabalho.

O projeto será avaliado em termos globais produzindo-se uma nota de projeto, que será distribuída por todos os elementos do grupo, caso estes tenham todos contribuído de igual forma para o projeto. Caso existam diferenças no nível de contribuição, então os estudantes que mais contribuíram poderão ter uma nota individual superior à nota de projeto e o inverso para os estudantes que menos contribuíram. No fundo, parte da nota individual dos estudantes que menos contribuíram pode ser transferida para os estudantes que mais contribuíram, recompensando assim o esforço adicional de quem mais contribuiu. Assim, **o grupo deverá indicar no seu relatório uma estimativa de qual a percentagem de contribuição de cada elemento do grupo para o conjunto das tarefas realizadas (entre 0% e 100%), por forma a aferir qual foi a contribuição individual de cada elemento do grupo**. É ainda requerido uma justificação sucinta do porquê das diferenças nas percentagens atribuídas aos vários elementos do grupo (caso essas diferenças existam). Assume-se que a informação prestada é consensual entre os vários elementos do grupo. Quando isso não for possível, tal facto deverá ficar expresso no documento.

ATENÇÃO: **é obrigatório** incluir também neste relatório uma **autoavaliação de cada membro do grupo** tendo em conta as três partes realizadas do trabalho.

Política em caso de fraude

Os alunos podem partilhar e/ou trocar ideias entre si sobre os trabalhos e/ou resolução dos mesmos. No entanto, o trabalho entregue deve corresponder ao esforço individual de cada grupo. São consideradas fraudes as seguintes situações:

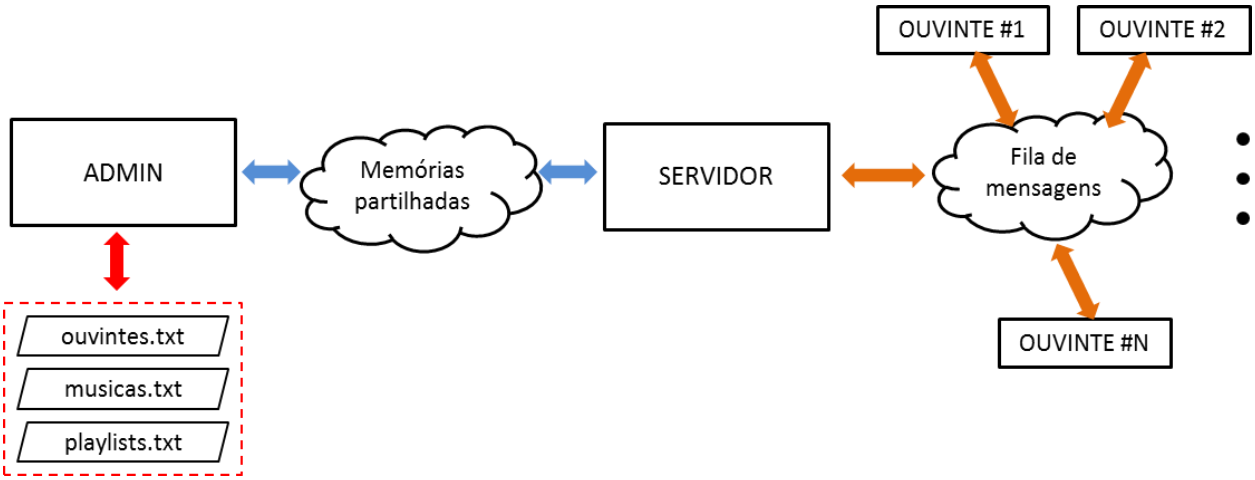
- Trabalho parcialmente copiado
- Facilitar a cópia através da partilha de ficheiros
- Utilizar material alheio sem referir a sua fonte.

Em caso de deteção de algum tipo de fraude, os trabalhos em questão não serão avaliados, sendo enviados à Comissão Pedagógica ou ao Conselho Pedagógico, consoante a gravidade da situação, que decidirão a sanção a aplicar aos alunos envolvidos. Serão utilizadas as ferramentas *Moss* e *SafeAssign* para deteção automática de cópias.

Recorda-se ainda que o Anexo I do Código de Conduta Académica, publicado a 25 de janeiro de 2016 em Diário da Republica, 2ª Série, nº 16, indica no seu ponto 2 que:
Quando um trabalho ou outro elemento de avaliação apresentar um nível de coincidência elevado com outros trabalhos (percentagem de coincidência com outras fontes reportada no relatório que o referido software produz), cabe ao docente da UC, orientador ou a qualquer elemento do júri, após a análise qualitativa desse relatório, e em caso de se confirmar a suspeita de plágio, desencadear o respetivo procedimento disciplinar, de acordo com o Regulamento Disciplinar de Discentes do ISCTE - Instituto Universitário de Lisboa, aprovado pela deliberação nº 2246/2010, de 6 de dezembro.
O ponto 2.1 desse mesmo anexo indica ainda que:
No âmbito do Regulamento Disciplinar de Discentes do ISCTE-IUL, são definidas as sanções disciplinares aplicáveis e os seus efeitos, podendo estas variar entre a advertência e a interdição da frequência de atividades escolares no ISCTE-IUL até cinco anos.

Parte 3 – Comunicação entre processos
Data de entrega: 14 de dezembro de 2019

Nesta parte do trabalho serão desenvolvidos os módulos **admin.c** e **servidor.c** (a correr no lado do servidor), e o módulo **ouvinte.c** (a correr no lado do cliente), utilizando a linguagem de programação C, nos quais se vão introduzir os mecanismos de comunicação entre processos: memórias partilhadas, semáforos e filas de mensagens. A figura seguinte representa o esquema geral de funcionamento do sistema.



A lista de ouvintes do sistema passará a estar disponível numa memória partilhada, que será utilizada tanto pelo programa **admin.c** como pelo **servidor.c**. A comunicação entre os vários processos de ouvinte e o servidor é feita através de uma fila de mensagens, através da qual serão feitos pedidos para diferentes tipos de operações, tais como: *login*, *listar músicas*, *ouvir playlist*, *associar a playlist* e *logout*. Os programas devem utilizar semáforos sempre que necessário, especialmente quando são utilizadas memórias partilhadas, por forma a garantir exclusão mútua e evitar colisões.

Os programas a desenvolver irão operar com os ficheiros **ouvintes.txt**, **musicas.txt** e **playlists.txt**. A estrutura de cada ficheiro é descrita em seguida.
Nota: os ficheiros de texto referidos serão fornecidos pelos docentes na diretoria do servidor Tigre **"/home/so/trabalho-parte-3"**, pelo que não será necessária a criação dos mesmos.

ouvintes.txt
Este ficheiro contém os ouvintes registados na aplicação Spotif-IUL, contendo as credenciais para login entre outras informações sobre cada ouvinte:
`nickname:password:numero_aluno:nome:email:curso:lista_de_playlists`

Exemplo:

```
Batman:robin:12345:João Caniço:a12345@iscte-iul.pt:IGE:PL_1:PL_2\n
asterix:obelix:67890:Marta Santos:a67890@iscte-iul.pt:ETI:PL_2\n
...
```

Observação: note que cada linha poderá ter número de argumentos variável, já que um ouvinte pode estar associado a mais do que uma playlist (pode até nem estar associado a nenhuma playlist).

musicas.txt

Este ficheiro guarda a lista de músicas que existe na aplicação. Deve seguir o seguinte formato (é, na prática, o mesmo formato da parte 2, mas agora o último campo da música corresponde ao número de vezes que essa música foi ouvida):

`ID_musica:nome_musica:intérprete:duração:ano_produção:género:num_vezes_ouvida`

Exemplo:

```
MUS_1:Bohemian Rhapsody:Queen:359:1975:rock:37\n
MUS_2:Ai se ele cai:Xutos e Pontapes:191:2004:rock:7\n
MUS_3:We are the champions:Queen:190:1977:rock:13\n
MUS_4:Estou alem:Antonio Variacoes:303:1983:rock:13\n
MUS_5:Like a prayer:Madonna:338:1989:pop:9\n
...
```

playlists.txt

Este ficheiro guarda as playlists registadas no sistema, ou seja, a lista ordenada de músicas que pertence a cada playlist. Deve seguir o seguinte formato (é, na prática, o mesmo formato das partes 1 e 2):

`ID_playlist:nome_playlist:lista_ordenada_de_identificadores_musicas`

Exemplo:

```
PL_1:Jogging:MUS_3:MUS_2\n
PL_2:Estudo:MUS_5:MUS_3:MUS_4\n
PL_3:Classicas:MUS_8:MUS_10:MUS_12:MUS_1:MUS_15\n
```

Nos programas a desenvolver, sugerem-se as seguintes estruturas para **ouvintes**, **músicas** e **playlists**. No entanto, os alunos podem-lhes adicionar/alterar campos da forma que necessitarem:

```
typedef struct {
      char nick[50];          // Nickname do ouvinte (para o Login)
      char pass[50];          // Password do ouvinte (para o Login)
      int num;                // Número de aluno
      char nome[50];          // Nome do ouvinte
      char email[50];         // E-mail
      char curso[50];         // Curso
      char playlists[250];    // Lista de playlists registadas separada por ":"
} Touvinte;
```

```
typedef struct {
      char id_mus[10];  // Identificador da música, de formato MUS_xxxxx
      char titulo[60];  // Título da Música
      char artista[50]; // Nome do intérprete
      int duracao;      // Duração da música (em segundos)
      int ano;          // Ano de lançamento da música
      char genero[20];  // Género musical
      int top;          // Posição da música no top
} Tmusica;
```

```
typedef struct {
      char id_pl[10];   // Identificador da playlist, de formato PL_xxxxx
      char nome[60];    // Nome da playlist
      char musicas[250];// Lista ordenada de músicas, separadas por ":"
} Tplaylist;
```

Nota: as listas identificadas nas estruturas Touvinte (lista de Playlists) e Tplaylist (lista de Músicas) são representadas por uma "string" com o formato já conhecido, como exemplos serão "PL_1:PL_6:PL_3" e "MUS_8:MUS_3:MUS_5".

Relativamente à comunicação entre o ouvinte e o servidor, recomendam-se as seguintes estruturas de dados, não sendo necessário especificar todos os campos em alguns pedidos. Os alunos poderão utilizar campos adicionais se for conveniente.

Cliente → Servidor	Servidor → Cliente

`typedef struct {`
`  long tipo;`
`  struct {`
`  char operacao[20];`
`  char info1[50];`
`  char info2[50];`
`  int myid;  // Numero Aluno`
`  } dados;`
`} MsgClient2Server;`
	`typedef struct {`
	`  long tipo;`
	`  struct {`
	`  Tmusica musica;`
	`  char info1[50];`
	`  int valor1;`
	`  int status;`
	`  } dados;`
	`} MsgServer2Client;`

1) Admin.c

Este programa é responsável por criar as memórias partilhadas e colocar lá os dados dos utilizadores. Ao correr, deve começar por verificar se a memória partilhada já existe. Caso exista, deve apenas associar-se a ela. Caso não exista, deverá criá-la.

Este programa é também o responsável por criar e inicializar os semáforos que serão usados para garantir exclusão mútua no acesso à memória partilhada. Deve começar por verificar seu grupo de semáforos já existe. Caso exista deve apenas associar-se a ele. Caso não exista deverá criá-los e iniciá-los adequadamente.

Deve também criar a fila de mensagens, necessária para a comunicação entre servidor e ouvintes.

De seguida deve apresentar um menu com as seguintes opções:

```
-------- MENU --------
1. Carregar ficheiros
2. Descarregar memória
3. Manutenção aplicação
4. Mostrar memória
0. Sair
```

- **Carregar ficheiros:** ler a informação sobre os ouvintes, músicas e playlists a partir dos respetivos ficheiros e carregar essa informação para estruturas de dados em memória (que devem ser previamente iniciadas com valores indicando "registo vazio"). Para cada estrutura deverá ser criada uma memória partilhada (Mem_musicas, Mem_playlists e Mem_ouvintes) e respetivo semáforo para controlo de acesso (Sem_musicas, Sem_playlists e Sem_ouvintes). **Recomenda-se que os semáforos sejam a última funcionalidade a ser implementada**.

- **Descarregar memória**: ler a informação das memórias partilhadas e atualizar os respetivos ficheiros musicas.txt, playlists.txt e ouvintes.txt.

- **Manutenção aplicação**: Apesar de neste trabalho não vir a ser desenvolvida nenhuma ação em concreto, pretende-se mimicar a necessidade de manutenção deste tipo de aplicações, durante a qual as memórias estarão inacessíveis. Quando esta funcionalidade for selecionada, o programa deve imprimir uma mensagem como se mostra abaixo e adormecer durante 10 segundos:

```
**********************************
            EM MANUTENÇÃO
**********************************
```

  Após os 10 segundos, escreve uma mensagem a dizer que a manutenção terminou e volta ao menu.

- **Mostrar memória**: apresenta o conteúdo de cada uma das memórias partilhadas de forma estruturada.

- **Sair**: desliga a interface do administrador. Salienta-se que as memórias, fila de mensagens e semáforos devem continuar disponíveis, apesar de o programa **admin** ser desligado.

2) Servidor.c

Este é um dos principais componentes da arquitetura do Spotif-IUL. É ele o responsável por dar resposta a todos os pedidos dos vários clientes. Deve operar os semáforos sempre que se justifique e implementar a resposta aos pedidos que chegam via fila de mensagens. Nomeadamente, deve garantir resposta a mensagens:

- **Pedido de autenticação**: deve validar o nickname e a password, com base na informação presente na memória partilhada dos ouvintes e responder ao ouvinte, confirmando, ou não, o login (por exemplo, utilizar a variável status como sendo booleana).

- **Listar músicas de playlist**: deve responder ao pedido do ouvinte ("listen_playlist") com várias mensagens, uma por cada música associada à playlist solicitada, com a informação sobre a mesma. Caso a playlist não seja válida, deve enviar uma mensagem de erro ao ouvinte (por exemplo, utilizar a variável status como sendo booleana).

- **Ouvir playlist**: deve responder ao pedido com várias mensagens, uma por cada música associada à playlist solicitada, com a informação sobre a mesma, e atualizar o campo correspondente ao número de vezes que as músicas foram ouvidas. Caso a playlist não seja válida, deve enviar uma mensagem de erro ao ouvinte (por exemplo, utilizar a variável status como sendo booleana).

- **Associar a playlist**: Responde ao pedido do ouvinte "get_playlist". Deve aceder à memória partilhada de playlists e verificar se a playlist, a que o ouvinte se quer associar, existe. Deve, também, aceder à memória partilhada dos ouvintes e verificar se o utilizador já está ou não associado à playlist pedida. Caso o ouvinte não esteja associado e a playlist exista, deve associar o utilizador à playlist, registando essa ação na memória partilhada dos ouvintes e responder, ao ouvinte, em conformidade.

Note que em qualquer acesso às memórias partilhadas, o servidor deve previamente verificar se a aplicação se encontra em manutenção (opção 3 do programa administrador). Se assim for, deve responder ao utilizador com essa informação e ignorar o pedido.

3) Ouvinte.c

Este programa é aquele que é usado pelos ouvintes para interagirem com a aplicação Spotif-IUL. Ao ser executado, deverá pedir o ***nickname*** e ***password*** ao utilizador e tentar fazer a sua autenticação, **enviando ao servidor uma mensagem** com o campo *operacao* preenchido com "login". A resposta do servidor é **recebida através de uma mensagem** cujo tipo é o PID do processo ouvinte. Caso a autenticação tenha sido **bem-sucedida**, o ouvinte deverá obter o seu número de aluno (*num*) como resposta, que será utilizado para futuras comunicações com o servidor. Nas mensagens de autenticação, o ouvinte envia o seu PID na mensagem (no campo myid) e ficará à espera de uma mensagem de resposta do servidor com o tipo igual ao seu PID. Na operação de login, o servidor reponde com o sucesso ou insucesso da operação. Após receber uma mensagem de autenticação com "sucesso", o programa ouvinte deve apresentar um menu com as seguintes opções:

```
-------- MENU --------
1. Listar músicas de playlist
2. Ouvir playlist
3. Associar a playlist
0. Sair (Logout)
```

- **Listar músicas de playlist**: envia pedido ao servidor preenchendo o campo *operacao* da estrutura de mensagem com "list_musics" e fica a aguardar a resposta do servidor.

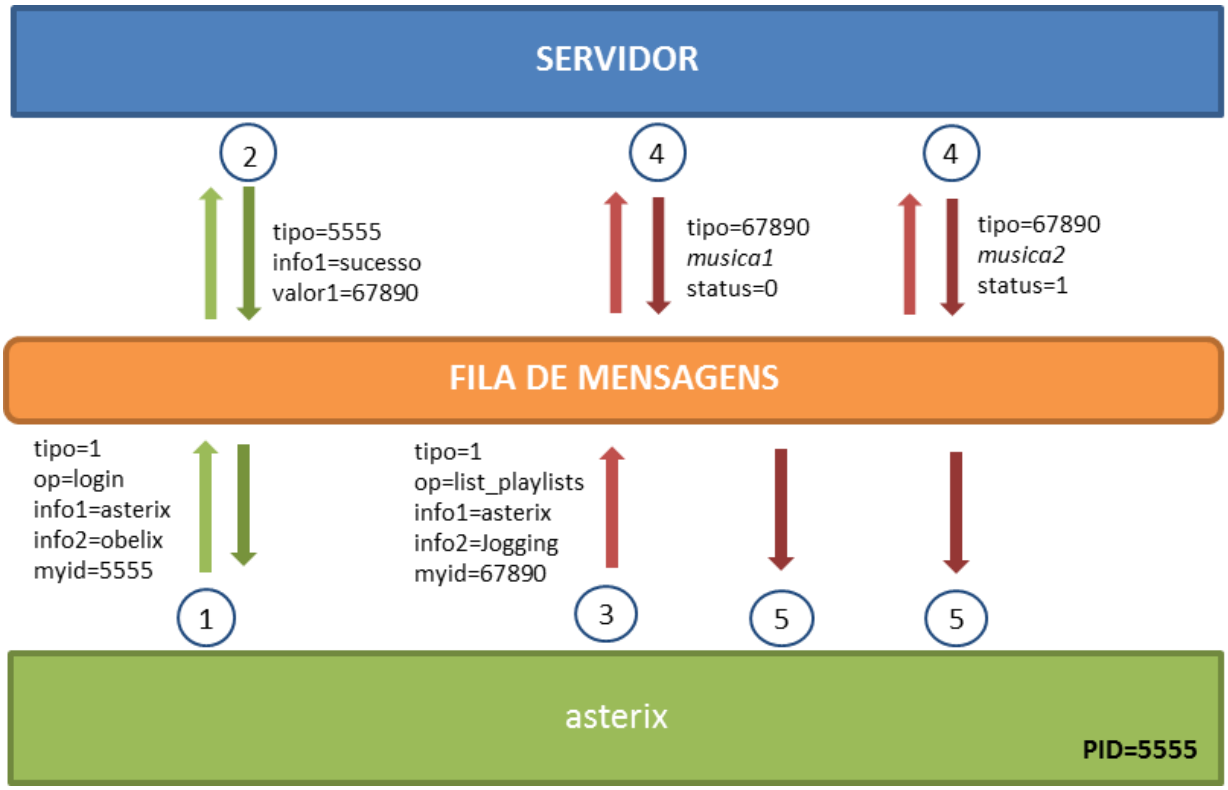
- **Ouvir playlist**: envia pedido ao servidor preenchendo os campos *operacao* e *info1* da estrutura de mensagem com "listen_playlist" e o nome da playlist, previamente solicitado ao utilizador, respetivamente. Após receber as mensagens do servidor, deve apresentar as playlists e respetivas músicas ao utilizador.

- **Associar a playlist**: envia pedido ao servidor preenchendo os campos *operacao* e *info1* com "get_playlist" e o nome da playlist, previamente solicitado ao utilizador, respetivamente. De seguida, fica a aguardar a mensagem de confirmação do servidor.

- **Sair (Logout)**: envia pedido ao servidor, preenchendo o campo *operacao* com "logout". Deve aguardar a resposta (sucesso/insucesso) do servidor.

Salienta-se que o servidor poderá indicar que a aplicação se encontra em manutenção. Nesta situação, o pedido do ouvinte não será executado e deverá ser dado feedback ao ouvinte para tentar novamente mais tarde.

A figura seguinte apresenta um exemplo da troca de mensagens entre o ouvinte e o servidor:



- As mensagens que o ouvinte envia ao servidor são sempre do **tipo 1**.

- A resposta do servidor a um pedido de autenticação é feita através de uma mensagem com o **tipo igual ao PID** do processo ouvinte.

- As restantes respostas do servidor são sempre do **tipo igual ao *num* (número de aluno)** desse ouvinte, que terá sido enviado ao ouvinte se a autenticação tiver sido bem-sucedida, o qual deve ser sempre preenchido no campo *myid*.