Sistemas Operativos 2023/2024

Trabalho Prático

Simulador de Sistema de Autorização de Serviços Móveis 5G

NOTA: Antes de começar o trabalho leia o enunciado até ao fim!

1. Introdução

A elevada velocidade e baixa latência fornecida pelas sucessivas gerações da rede móvel, nomeadamente o 4G e mais recentemente o 5G, tem motivado o consumo exponencial de serviços móveis pelos utilizadores na última década. Os serviços de dados (p. ex., redes sociais, *streaming* vídeo, *streaming* música, etc.) têm hoje uma enorme utilização. Os operadores de telecomunicações, para além de garantirem o fornecimento destes serviços com qualidade, têm também que autorizar a utilização dos mesmos em tempo-real de acordo com o plafond de cada cliente. Um operador de telecomunicações faz este controlo através de plataformas de autorização em tempo-real (*authorization engines*), com elevada disponibilidade para responder a milhares de pedidos de serviço em simultâneo, garantindo assim que os utilizadores não têm que esperar para aceder aos serviços. Se a resposta a um pedido de serviço ultrapassar algumas centenas de milissegundos, o serviço não se inicia, o que contribui para a insatisfação do cliente.

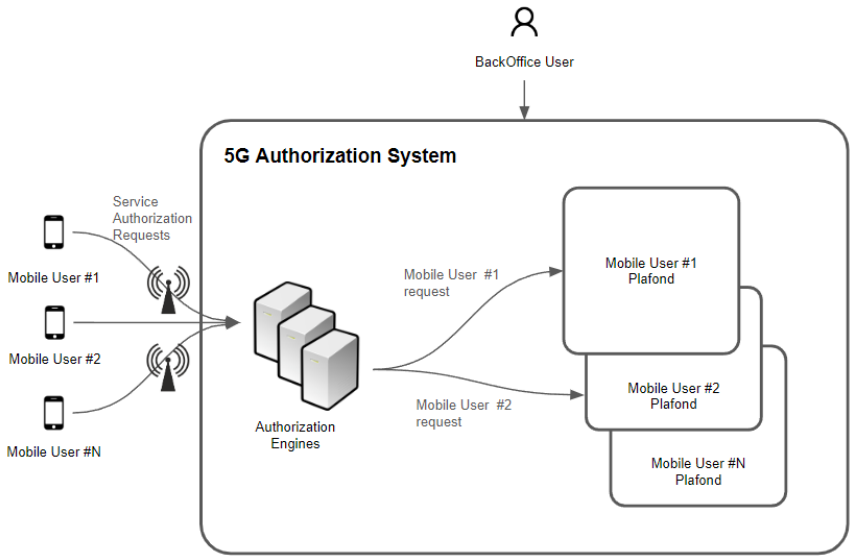

Figura 1: Simulador de Sistema de Autorização de Serviços Móveis 5G

O trabalho a desenvolver neste projeto deve simular um sistema de autorização em tempo real onde vários utilizadores móveis tentam utilizar vários serviços de dados em simultâneo, conforme apresentado na Figura 1. O sistema deve receber os pedidos de utilização de serviço e autorizar os mesmos de acordo com o plafond disponível de cada cliente. Para além do processo de autorização, devem também ser geradas estatísticas agregadas de utilização, bem como alertas de consumo do plafond para os utilizadores.

2. Descrição do sistema de simulação

Nesta secção será descrita a arquitetura técnica do sistema, incluindo as funcionalidades dos vários componentes do simulador e a forma como comunicam.

2.1. Estrutura do sistema

A arquitetura técnica do simulador está representada na Figura 1.

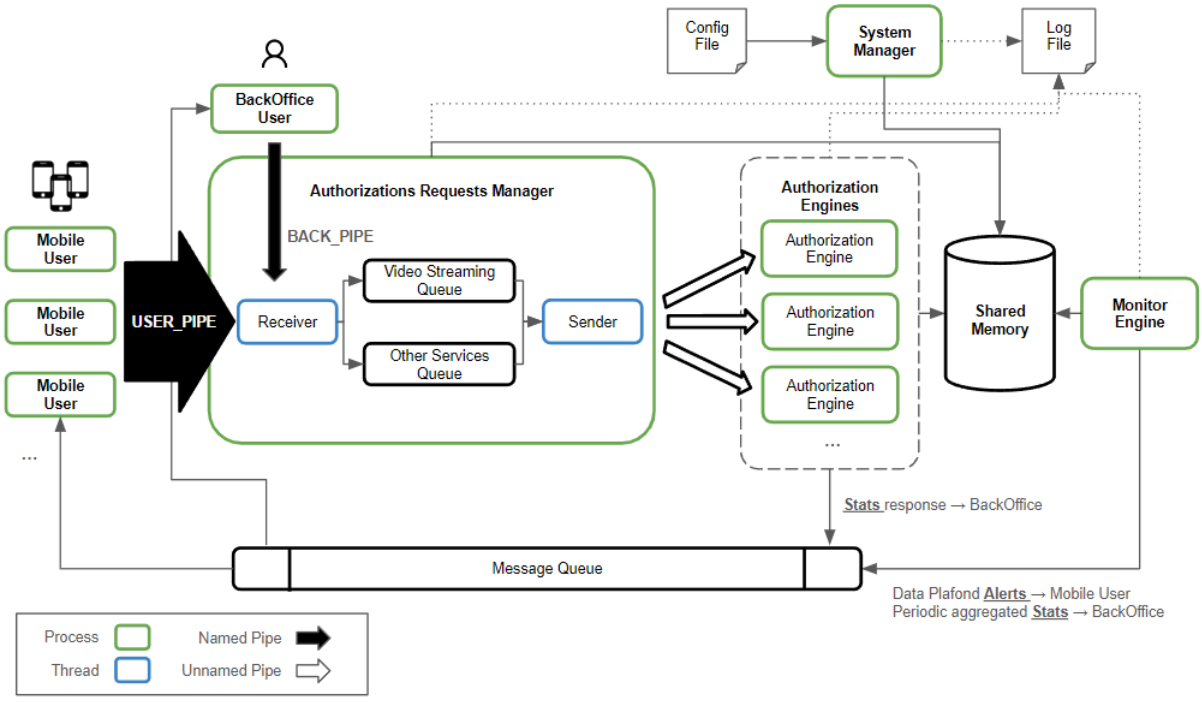

Figura 2: Arquitetura técnica do simulador

O sistema é baseado nos seguintes processos e *threads*:

- ***Mobile User*** - Processo que simula o utilizador móvel e gera pedidos de serviço de dados (redes sociais, vídeo ou música) utilizando um *named pipe* comum (**USER_PIPE**) para comunicar com o ***Authorization Requests Manager***. É também responsável pela receção de alertas relacionados com os consumos dos vários serviços (via ***Message Queue***). Podem existir vários destes processos.
- ***BackOffice User*** - Processo que gere informação agregada da utilização dos serviços pelos utilizadores. Recebe periodicamente estatísticas agregadas de utilização dos serviços (via ***Message Queue***), ou pode também solicitar estatísticas de forma assíncrona ao ***Authorization Requests Manager*** utilizando um *named pipe* (**BACK_PIPE**). Existe, no máximo, apenas um destes processos.
- ***System Manager*** - Processo responsável por iniciar o sistema, ler o ficheiro de configuração, criar a ***Shared Memory***, criar a ***Message Queue***, e criar os processos ***Authorization Requests Manager*** e ***Monitor Engine***.

- ***Authorization Requests Manager*** - Processo responsável por gerir a receção de pedidos do ***Mobile User*** e do ***BackOffice User***, encaminhar os mesmos para a respetiva fila interna (***Video Streaming Queue***, ***Other Services Queue***) e a sua distribuição pelos processos ***Authorization Engine*** de acordo com as prioridades definidas. Responsável pela criação dos processos ***Authorization Engine***, de acordo com a taxa de ocupação das filas. É também responsável pela criação dos *named pipes*.
- ***Authorization Engine*** - Processo responsável por executar pedidos de autorização de serviço. Sempre que recebe um pedido de autorização de serviço, deve verificar se o utilizador tem plafond disponível para prosseguir com o mesmo e fazer a sua contabilização na ***Shared Memory***. É também responsável por responder à solicitação de estatísticas do ***BackOffice User***. Podem existir vários destes processos.
- ***Monitor Engine*** - Processo responsável por gerar alertas se os valores dos consumos de cada utilizador atingem os limites definidos pelo sistema (80%, 90%, 100%). Se for este o caso, então deve alertar os mesmos através da ***Message Queue***. É também responsável por enviar as estatísticas periódicas agregadas para o ***BackOffice User***.
- ***Receiver*** - *Thread* que lê os comandos do ***Mobile User*** e do ***BackOffice User*** enviados através dos *named pipes*.
- ***Sender*** - *Thread* que lê os pedidos que estão armazenados nas filas (***Video Streaming Queue***, ***Other Services Queue***) e envia os mesmos através de *unnamed pipes* para um ***Authorization Engine*** que esteja disponível. Os pedidos de autorização na fila ***Video Streaming Queue*** têm prioridade sobre os pedidos na fila ***Other Services Queue***.

Para além dos processos e *threads* descritos acima, o sistema é também constituído por vários IPCs:

- ***Named Pipe*** - Permite que os ***Mobile Users*** (**USER_PIPE**) e o ***BackOffice User*** (**BACK_PIPE**) enviem dados ao processo ***Authorization Requests Manager***.
- **SHM** - Zona de memória partilhada acedida pelos processos ***System Manager***, ***Authorization Engine, Authorization Request Manager*** e ***Monitor Engine***.
- ***Unnamed pipes*** - Permitem a comunicação entre o processo ***Authorization Requests Manager*** e cada um dos processos ***Authorization Engine***.
- ***Message Queue*** (fila de mensagens) - Permite o envio (pelo ***Authorization Engine***) da resposta aos pedidos de autorização dos ***Mobile Users***, bem como o envio de estatísticas agregadas solicitadas pelo ***BackOffice User***. Permite também o envio de alertas a partir do ***Monitor Engine*** para os ***Mobile Users***.

Dentro do processo ***Authorization Requests Manager*** existem duas estruturas de dados de tamanho fixo que armazenam os dados e comandos que têm de ser processados pelos ***Authorization Engines***:

1. ***Video Streaming Queue***: fila que armazena pedidos de autorização do serviço de vídeo streaming enviados pelos ***Mobile Users***;
2. ***Other Services Queue***: fila que armazena todos os outros pedidos de autorização de serviço enviados pelos ***Mobile Users*** e os comandos enviados pelo ***BackOffice User***;

Existirá também um ficheiro de ***log*** onde todos os processos deverão escrever as informações relevantes sobre a sua atividade para análise posterior. Todas as informações escritas para o ***log*** também devem aparecer no ecrã.

2.2. Descrição dos componentes e das funcionalidades

De seguida são apresentadas as características e funcionalidades dos diversos componentes.

Mobile User

Processo que gera pedidos de autorização para cada um dos 3 serviços do simulador (*streaming* de vídeo, *streaming* de música e redes sociais). O ***Mobile User*** gera duas mensagens:

1. **Registo inicial**: mensagem inicial para simular o registo do ***Mobile User*** na plataforma de autorizações de serviço. Neste pedido terá que ser indicado o plafond inicial do ***Mobile User***. Este valor é registado na ***Shared Memory***.
2. **Pedido de autorização**: mensagem para simular os pedidos de autorização de serviço do ***Mobile User***. Estas mensagens são enviadas em **intervalos periódicos (Δt)**, específicos para cada tipo de serviço. Para cada pedido de autorização é indicada **a quantidade de dados a reservar** do plafond disponível. Este passo repete-se até o **número máximo de pedidos de autorização** estar concluído ou o plafond esgotado.

A Figura 3 exemplifica o funcionamento do ***Mobile User***.

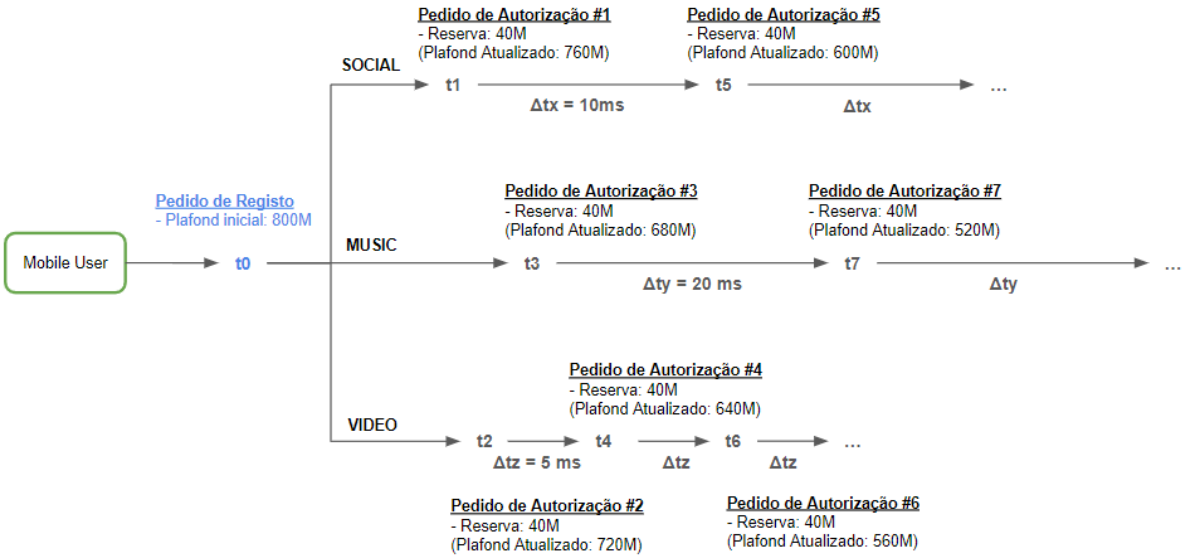

Figura 3: Funcionamento do ***Mobile User***

O **plafond inicial**, o **número de pedidos de autorização** a enviar, os **intervalos periódicos de renovação (Δt) por serviço** e a **quantidade de dados a reservar** em cada pedido de renovação é fornecido através da linha de comandos no arranque do ***Mobile User***.

Cada um dos processos ***Mobile User*** envia as mensagens através do *named pipe* **USER_PIPE**. Podemos ter um ou mais processos destes a correr em simultâneo, cada um com os seus parâmetros.

Sintaxe do comando de inicialização do processo ***Mobile User***:

```
$ mobile_user /
{plafond inicial} /
{número de pedidos de autorização} /
{intervalo VIDEO} {intervalo MUSIC} {intervalo SOCIAL} /
{dados a reservar}
```

Exemplo:

```
$ mobile_user 800 50 10 20 5 40
```

Informação a enviar para o *named pipe* para a **mensagem de registo**:

```
ID_mobile_user#Plafond inicial
```

Exemplo:
345#800

Informação a enviar para o *named pipe* relativamente à **mensagem de pedido de autorização** (para cada um dos 3 serviços):

```
ID_mobile_user#ID serviço#Quantidade de dados a reservar
```

Exemplo (para cada serviço):
345#VIDEO#40
345#MUSIC#40
345#SOCIAL#40

O identificador do ***Mobile_User***, correspondente ao PID, será utilizado para agrupar a informação do utilizador na memória partilhada.

O ***Mobile_User*** recebe alertas sobre o plafond de dados (80%, 90%, 100%) através da ***Message Queue***.

O processo ***Mobile User*** termina quando uma das seguintes condições se verificar:

1. Receção de um **sinal SIGINT**;
2. Receção de um **alerta de 100%** relativo ao plafond de dados;
3. No caso de o **número máximo de pedidos de autorização** ser atingido;
4. Em caso de erro - um erro pode acontecer se algum parâmetro estiver errado ou ao tentar escrever para o *named pipe* e a escrita falhar. Nestes casos deve escrever a mensagem de erro no ecrã.

Sempre que o ***Mobile User*** termina, o processo deve limpar todos os recursos.

BackOffice User

Processo que gere informação agregada dos plafonds dos utilizadores. Recebe estatísticas periódicas (produzidas pelo ***Monitor Engine***) através da ***Message Queue***. Pode também, proactivamente, solicitar estatísticas utilizando um comando específico. Neste caso, o comando é enviado para o ***Authorization Request Manager*** através do *named pipe* **BACK_PIPE**. O ***Authorization Engine*** é responsável por processar as estatísticas e enviar as mesmas ao ***BackOffice User*** através da ***Message Queue***.

Sintaxe do comando de inicialização do processo ***BackOffice User***:

```
$ backoffice_user
```

Exemplo:

```
$ backoffice_user
```

Informação a enviar para o *named pipe*:

```
ID_backoffice_user#[data_stats | reset]
```

O identificador do ***BackOffice User*** a utilizar é 1.
Este processo recebe os seguintes comandos do utilizador:

- **data_stats** - apresenta estatísticas referentes aos consumos dos dados nos vários serviços: total de dados reservados e número de pedidos de renovação de autorização;
- **reset** - limpa as estatísticas relacionadas calculadas até ao momento pelo sistema.

Exemplo da utilização do comando:

```
> 1#data_stats
Service     Total Data  Auth Reqs
VIDEO       130000      3000
MUSIC       42000       2100
SOCIAL      790000      8500

> 1#reset
```

O processo termina ao receber um **sinal SIGINT**, ou em caso de erro. Um erro pode acontecer se algum parâmetro estiver errado ou ao tentar escrever para o *named pipe* e a escrita falhar, casos em que deverá escrever a mensagem de erro no ecrã. Sempre que termina, o processo deve limpar todos os recursos.

System Manager

Este processo lê as configurações iniciais e arranca todo o sistema. Em concreto tem as seguintes funcionalidades:

- Lê e valida as informações no ficheiro de configurações - ***Config File*** (neste enunciado é fornecido um exemplo deste ficheiro)
- Cria os processos ***Authorization Requests Manager e Monitor Engine***
- Cria a ***Message Queue***
- Cria a ***Shared Memory***;
- Escreve no log file;
- Captura o sinal ***SIGINT*** para terminar o programa, libertando antes todos os recursos.

Sintaxe do comando:

```
$ 5g_auth_platform {config-file}
```

Authorization Requests Manager

Este processo é responsável por receber os dados do ***Mobile User*** e do ***BackOffice User***. Coloca a informação recebida numa de duas filas internas, e distribui os pedidos pelos processos ***Authorization Engine*** de acordo com as prioridades definidas. Em concreto tem as seguintes funcionalidades:

- Cria os *named pipes* **USER_PIPE** e **BACK_PIPE**
- Cria os *unnamed pipes* para cada ***Authorization Engine***
- Cria as *threads* ***Receiver*** e ***Sender***
- Cria as estruturas de dados internas: **VIDEO_STREAMING_QUEUE** e **OTHERS_QUEUE**
- Criação e remoção dos processos ***Authorization Engine*** de acordo com a taxa de ocupação das filas.

Receiver

Esta *thread* é responsável por receber os pedidos provenientes do ***Mobile User*** e do ***BackOffice User*** através dos respectivos *named pipes* (**USER_PIPE** e **BACK_PIPE**). A monitoria dos *file descriptors* dos *named pipes* terá de ser efetuada através da *system call* ***select()***, permitindo assim uma espera eficiente entre os múltiplos *file descriptors*. Os pedidos de autorização associados ao serviço de vídeo são colocados na fila ***VIDEO_QUEUE***, enquanto os restantes pedidos são colocados na fila ***OTHER_QUEUE***. Ambas as filas têm tamanho máximo de QUEUE_POS, sendo este parâmetro fornecido pelo ficheiro de configurações. Se a fila estiver cheia, o pedido é eliminado e essa informação é escrita no ecrã e no ficheiro de ***log***.

Sender

Esta *thread* é responsável por ler pedidos de autorização presentes nas filas de mensagens e entregar os mesmos ao ***Authorization Engine*** disponível. A fila de pedidos de autorização de vídeo (***VIDEO_STREAMING_QUEUE***) é prioritária relativamente à fila ***OTHER_SERVICES_QUEUE***. As seguintes considerações devem ser tidas em conta na implementação desta *thread*:

- Esta *thread* é notificada quando um novo pedido de autorização é colocado numa das filas. Nesse momento terá que selecionar um ***Authorization Engine*** que esteja disponível e entregar o pedido de autorização para execução.
- Sempre que uma das filas estiver cheia deve ser criado um ***Authorization Engine*** **extra**. A remoção do ***Authorization Engine*** **extra** deve ser efetuada quando a fila atingir os 50%.
- O número máximo de ***Authorization Engines*** é dado por AUTH_SERVERS_MAX, sendo este parâmetro fornecido pelo ficheiro de configurações.
- O tempo de processamento de cada ***Authorization Engine*** para processar a tarefa solicitada é dado por AUTH_PROC_TIME;
- Os pedidos de autorização de vídeo têm MAX_VIDEO_WAIT para serem atendidos. Os pedidos de autorização colocados na fila menos prioritária têm MAX_OTHERS_WAIT para serem atendidos. Se o pedido não for atendido no respetivo tempo limite, o mesmo é eliminado pela *thread* ***Sender*** e essa informação é escrita no ecrã e no ficheiro de ***log***.
- A comunicação entre o ***Sender*** e o ***Authorization Engine*** é feita através do *unnamed pipe* criado para cada ***Authorization Engine***. Quando o ***Authorization Engine*** estiver a processar

um pedido de autorização deve passar a estar no estado ocupado (para que o ***Sender*** não lhe envie outra mensagem enquanto esta não tiver sido processada).

Authorization Engine

Este processo é responsável por executar pedidos de autorização entregues pela *thread* ***Sender*** através do *unnamed pipe*. As mensagens recebidas pelo ***Authorization Engine*** são as seguintes:

- **Registo inicial**: mensagem inicial para simular o registo do ***Mobile User*** na plataforma de autorizações de serviço. O plafond inicial do ***Mobile User*** é indicado nesta mensagem e é registado na ***Shared Memory***. O plafond inicial de cada ***Mobile User*** pode ser distinto.
- **Pedido de autorização**: enviado no início da utilização de cada serviço, e em intervalos periódicos, contém a quantidade de dados a reservar para o serviço. O ***Authorization Engine*** deve verificar se o ***Mobile User*** ainda tem plafond disponível na ***Shared Memory***. Em caso afirmativo deve subtrair o valor da reserva ao plafond disponível. Se não tiver plafond suficiente, o pedido de autorização deve ser rejeitado e essa informação escrita no ecrã e no ficheiro de ***log***. Para o cálculo das estatísticas deve ser mantido o valor total dos dados solicitados para cada tipo de serviço, bem como o número total de pedidos que foram recebidos para cada tipo de serviço.
- **Pedido de estatísticas**: comando enviado pelo utilizador do ***BackOffice*** a solicitar estatísticas. O ***Authorization Engine*** deve aceder à ***Shared Memory***, ler os dados necessários e devolvê-los ao ***BackOffice User*** através da *Message Queue*.
- **Reset de estatísticas**: comando enviado pelo utilizador do ***BackOffice*** a solicitar o reset das estatísticas. O ***Authorization Engine*** deve reiniciar as estatísticas na ***Shared Memory***.

Monitor Engine

Este processo verifica quando é que o plafond de dados de cada ***Mobile User*** atinge 80%, 90% ou 100% do plafond inicial. Como resultado, são gerados alertas para os respetivos ***Mobile Users*** através da ***Message Queue***. Para além dos alertas, este processo é também responsável por gerar estatísticas periódicas (em intervalos de 30 segundos) para o ***BackOffice User***.

Para todos os processos do simulador

Todos os processos mantêm a memória partilhada atualizada, usando os mecanismos necessários para que não seja possível a existência de corrupção de dados. Todos os processos devem escrever para o ***log*** e para o ecrã as atividades que estão a executar.

Fim controlado do simulador de *offloading*

O sistema que controla e executa pedidos de autorização tem de terminar de forma controlada. Ao receber um SIGINT através do ***System Manager*** o sistema segue as seguintes etapas:

- Escreve no *log* que o programa vai acabar.
- As *threads* ***Receiver*** e ***Sender*** terminam a sua execução, deixando assim o sistema de receber dados dos processos ***Mobile User*** ou ***BackOffice User***.
- Aguarda que todos os pedidos de autorização e comandos que estejam a executar nos ***Authorization Engines*** terminem.
- Escreve no *log* as tarefas que estão nas filas e que não chegaram a ser executadas.

- Após as tarefas terminarem, remove todos os recursos utilizados pelo sistema (inclui todos os recursos utilizados, com exceção dos processos ***Mobile User*** e ***BackOffice User***).

Ficheiro de Configurações

O ficheiro de configurações deverá seguir a seguinte estrutura:

```
MOBILE_USERS – número máximo de Mobile Users que serão suportados pelo simulador
QUEUE_POS – número de slots nas filas que são utilizadas para armazenar os pedidos de autorização e os comandos dos utilizadores (>=0)
AUTH_SERVERS_MAX – número máximo de Authorization Engines que podem ser lançados (>=1)
AUTH_PROC_TIME – período (em ms) que o Authorization Engine demora para processar os pedidos
MAX_VIDEO_WAIT – tempo máximo (em ms) que os pedidos de autorização do serviço de vídeo podem aguardar para serem executados (>=1)
MAX_OTHERS_WAIT – tempo máximo (em ms) que os pedidos de autorização dos serviços de música e de redes sociais, bem como os comandos podem aguardar para serem executados (>=1)
```

Exemplo do ficheiro de configurações:

```
50
30
6
500
100
300
```

Log da aplicação

Todo o *output* da aplicação deve ser escrito de forma legível num ficheiro de texto "log.txt". Cada escrita neste ficheiro deve ser sempre precedida pela escrita da mesma informação na consola, de modo a poder ser facilmente visualizada enquanto decorre a simulação.

Deverá pôr no ***log*** todos os eventos relevantes acompanhados da sua data e hora, incluindo:

- Início e fim do programa;
- Criação de cada um dos processos e *threads*
- Erros ocorridos
- Pedidos de autorização descartados
- Mudança de estado de cada ***Authorization Engine***
- Alertas enviados pelo ***Monitor Engine***
- Sinais recebidos

Exemplo do ficheiro de *log*:

```
18:00:05 5G_AUTH_PLATFORM SIMULATOR STARTING
18:00:06 PROCESS SYSTEM_MANAGER CREATED
18:00:06 PROCESS AUTHORIZATION_REQUEST_MANAGER CREATED
18:00:06 THREAD RECEIVER CREATED
18:00:06 THREAD SENDER CREATED
18:00:06 PROCESS MONITOR_ENGINE CREATED
(...)
18:00:23 AUTHORIZATION_ENGINE 1 READY
18:00:23 AUTHORIZATION_ENGINE 2 READY
(…)
18:04:00 SENDER: VIDEO AUTHORIZATION REQUEST (ID = 345) SENT FOR PROCESSING ON AUTHORIZATION_ENGINE 1
18:04:00 SENDER: MUSIC AUTHORIZATION REQUEST (ID = 5011) SENT FOR PROCESSING ON AUTHORIZATION_ENGINE 2

(...)
18:04:30 AUTHORIZATION_ENGINE 1:VIDEO AUTHORIZATION REQUEST (ID = 345) PROCESSING COMPLETED
18:05:30 AUTHORIZATION_ENGINE 2: VIDEO AUTHORIZATION REQUEST (ID = 5011) PROCESSING COMPLETED

(...)
```

```
18:06:00 ALERT 80% (USER 1) TRIGGERED

(...)
18:05:50 SIGNAL SIGINT RECEIVED
18:06:00 5G_AUTH_PLATFORM SIMULATOR WAITING FOR LAST TASKS TO FINISH
18:06:10 5G_AUTH_PLATFORM SIMULATOR CLOSING
```

3. Checklist

Esta lista serve apenas como indicadora das tarefas a realizar e assinala as componentes que serão objeto de avaliação na defesa intermédia. Tarefas com "(preliminar)" não precisam de estar completas na defesa intermédia.

Item	Tarefa	Avaliado na defesa intermédia?
Mobile e *Backoffice User*	Criação dos processos *Mobile* (podem ser vários) e *Backoffice* (apenas 1). Estes processos são independentes.	S
	Leitura correta dos parâmetros de linha do comando e de comandos do utilizador	S
	Geração e escrita dos pedidos nos *named pipe* USER_PIPE e BACK_PIPE	
System Manager	Arranque do sistema, leitura do ficheiro de configurações, validação dos dados do ficheiro e aplicação das configurações lidas.	S
	Criação dos processos *Authorization Request Manager* e *Monitor Engine*	S
	Criação da memória partilhada	S
	Criação da fila de mensagens	
	Capturar o sinal SIGINT, terminar a corrida e liberta os recursos	
Authorization Requests Manager	Criação dos *named pipes* USER_PIPE e BACK_PIPE	
	Criação das *threads Receiver* e *Sender*	S
	Receção de pedidos do *Mobile* e *Backoffice user* e colocação nas filas corretas	
	Envio dos pedidos para os *Authorization Engines* por ordem de prioridade	
	Gestão dinâmica do *Authorization Engine* extra	
Authorization Engine	Gerir registos e pedidos de autorização	
	Gerir pedidos de estatísticas e *reset* das mesmas	
Monitor Engine	Gerar alertas	
Ficheiro *log*	Envio sincronizado do *output* para ficheiro de *log* e ecrã.	S
Geral	Criar um *makefile*	S
	Diagrama com a arquitetura e mecanismos de sincronização	S (preliminar)
	Suporte de concorrência no tratamento de pedidos	
	Deteção e tratamento de erros.	
	Atualização da shm por todos os processos e *threads* que necessitem	
	Sincronização com mecanismos adequados (semáforos, *mutexes* ou variáveis de condição)	S (preliminar)
	Prevenção de interrupções indesejadas por sinais não especificados; fornecer a resposta adequada aos vários sinais especificados no enunciado	
	Após receção de SIGINT, terminação controlada de todos os processos e *threads,* e libertação de todos os recursos.	

4. Notas importantes

- Leia atentamente este enunciado e esclareça dúvidas com os docentes.
- Em vez de começar a programar de imediato pense com tempo no problema e estruture adequadamente a sua solução. Soluções mais eficientes e que usem menos recursos serão valorizadas.
- Inclua na sua solução o código necessário à deteção e correção de erros.
- Evite esperas ativas no código, sincronize o acesso aos dados sempre que necessário e assegure a terminação limpa do servidor, ou seja, com todos os recursos utilizados a serem removidos.
- **Penalizações:**
  - A não compilação do código enviado implica uma classificação de **ZERO valores** na meta correspondente.
  - Esperas ativas serão fortemente penalizadas! Use o comando *top* num terminal, de modo a assegurar que o programa não gasta mais CPU que o necessário!
  - Acessos concorrentes que, por não serem sincronizados, puderem levar à corrupção de dados, serão fortemente penalizados!
  - Uso da função *sleep* ou estratagemas similares para evitar problemas de sincronização, serão fortemente penalizados!
- Inclua informação de *debug* que facilite o acompanhamento da execução do programa, utilizando por exemplo a seguinte abordagem:

```
#define DEBUG //remove this line to remove debug messages
(...)
#ifdef DEBUG
printf("Creating shared memory\n");
#endif
```

- Todos os trabalhos **deverão funcionar na VM fornecida** ou, em alternativa, na máquina **student2.dei.uc.pt**.
- Compilação: o programa deverá compilar com recurso a uma *makefile*; não deve ter erros em qualquer uma das metas; evite também os *warnings*, exceto quando tiver uma boa justificação para a sua ocorrência (é raro acontecer!).
- A defesa final do trabalho é obrigatória e todos os elementos do grupo devem participar. A não comparência na defesa final implica a classificação de **ZERO valores** no trabalho.
- O trabalho pode ser realizado em grupos de até 2 alunos (grupos com apenas 1 aluno devem ser evitados e grupos com mais de 2 alunos não são permitidos).
- Os alunos do grupo devem pertencer a turmas PL do mesmo docente. Grupos com alunos de turmas de docentes diferentes são exceções que carecem de aprovação prévia.
- A nota da defesa é individual pelo que cada um dos elementos do grupo poderá ter uma nota diferente;
- Ambas as defesas, intermédia e final, devem ser realizadas na mesma turma e com o mesmo docente.
- **Não será tolerado plágio, cópia de partes de código entre grupos ou qualquer outro tipo de fraude**. Tentativas neste sentido resultarão na **classificação de ZERO valores** e na consequente **reprovação na cadeira**. Dependendo da gravidade poderão ainda levar a processos disciplinares.
- Todos os trabalhos serão escrutinados para deteção de cópias de código.
- Para evitar cópias, os alunos **não podem** colocar código em repositórios de acesso público.

5. Metas, entregas e datas

Data	**Meta**	
Data de entrega no Inforestudante 02/04/2024-09h00	Entrega intermédia	Crie um arquivo no formato **ZIP** (**NÃO SERÃO ACEITES OUTROS FORMATOS**) com todos os ficheiros do trabalho e submeta-o no Inforestudante: ● Os **nomes** e **números** dos alunos do grupo devem ser colocados no início dos ficheiros com o código fonte. ● Inclua todos os ficheiros fonte e de configuração necessários e também um Makefile para compilação do programa. ● Não inclua quaisquer ficheiros não necessários para a compilação ou execução do programa (ex. diretórios ou ficheiros de sistemas de controlo de versões) ● Com o código deve ser entregue **1 página A4 com a arquitetura e todos os mecanismos de sincronização a implementar descritos**. ● **Não serão admitidas entregas por e-mail**.
Semana de 02/04/2024	Demonstração/ defesa intermédia	● A demonstração deverá contemplar todos os pontos referidos na *checklist* que consta deste enunciado. ● A demonstração/defesa será realizada nas aulas PL. ● A defesa intermédia vale **20%** da cotação do projeto.
Data de entrega final no Inforestudante 13/05/2024-09h00	Entrega final	Crie um arquivo no formato **ZIP** (**NÃO SERÃO ACEITES OUTROS FORMATOS**) com todos os ficheiros do trabalho e submeta-o no Inforestudante: ● Os **nomes** e **números** dos alunos do grupo devem ser colocados no início dos ficheiros com o código fonte. ● Inclua todos os ficheiros fonte e de configuração necessários e também um Makefile para compilação do programa. ● Não inclua quaisquer ficheiros não necessários para a compilação ou execução do programa (ex. diretórios ou ficheiros de sistemas de controlo de versões) ● Com o código deve ser entregue um **relatório** sucinto (no máximo 2 páginas A4), no formato **pdf (NÃO SERÃO ACEITES OUTROS FORMATOS)**, que explique as opções tomadas na construção da solução. Inclua um esquema da arquitetura do seu programa. Inclua também informação sobre o tempo total despendido (por cada um dos dois elementos do grupo) no projeto. ● **Não serão admitidas entregas por e-mail**.
13/05/2024 a 06/06/2024	Defesa final	● A defesa final vale **80%** da cotação do projeto e consistirá numa análise detalhada do trabalho apresentado. ● Defesas em grupo nas aulas PL. ● É necessária inscrição para a defesa.

Depois da submissão é aconselhado o *download* dos ficheiros, para verificar se tudo o que é necessário foi submetido. Caso submeta a pasta errada, apenas parte dos ficheiros, etc., isso não poderá contar para a meta.