

## Trabalho Prático

### Simulador para *Offloading* de Tarefas no *Edge*

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

#### 1. Introdução

Em ciência da computação, *offloading* computacional refere-se à transferência de processamentos computacionais para uma plataforma externa, como um cluster de computadores, *grid* computacional, ou uma nuvem. O *offloading* pode ser necessário devido às limitações de *hardware* e energia nos dispositivos clientes (e.g., telemóveis). Essas computações intensivas podem ser usadas em inteligência artificial, visão computacional, rastreamento de objetos, sistemas de suporte à decisão, *big data*, entre outros.

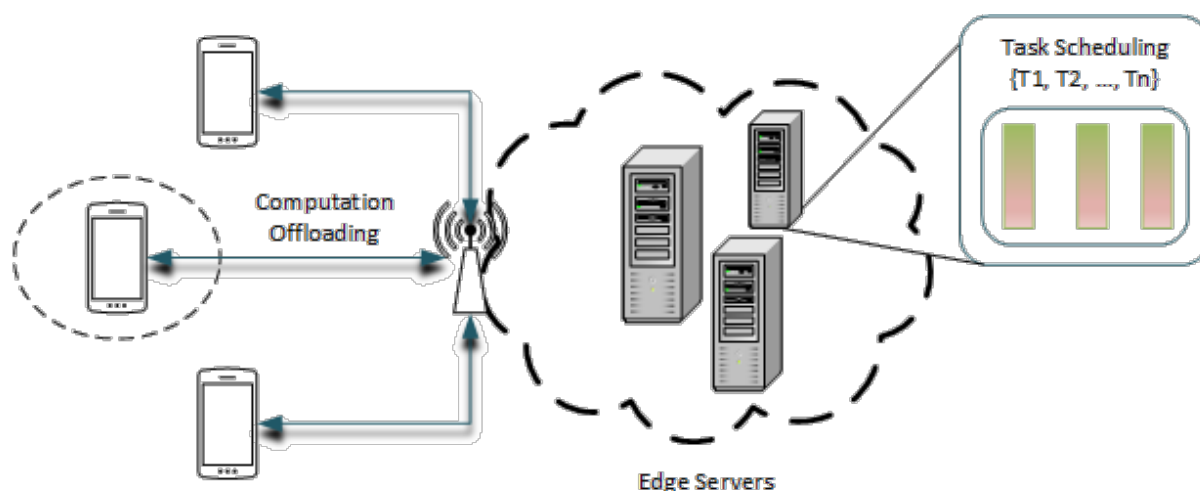


Figura 1. *Offloading* de tarefas computacionais no *edge*.

Tendo em conta a definição de *offloading* descrita anteriormente, o trabalho a desenvolver deve simular um ambiente simplificado de *edge computing* onde vários clientes podem delegar (fazer *offload*) tarefas para nós (servidores) que se encontram geograficamente próximos do cliente. Para distribuir os pedidos dos clientes pelos recursos existentes é necessário um sistema que mantenha informação sobre os recursos disponíveis e a sua utilização, que receba os pedidos com as tarefas que os clientes precisam de ver satisfeitas e que distribua essas tarefas pelos vários servidores. De seguida são descritos os detalhes do sistema simplificado que deve ser simulado neste trabalho.

#### 2. Descrição do sistema de simulação

Nesta secção serão descritos os componentes do sistema, as suas funcionalidades e a forma como comunicam entre si.

##### 2.1. Estrutura do sistema

A estrutura do sistema é apresentada na Figura 2.

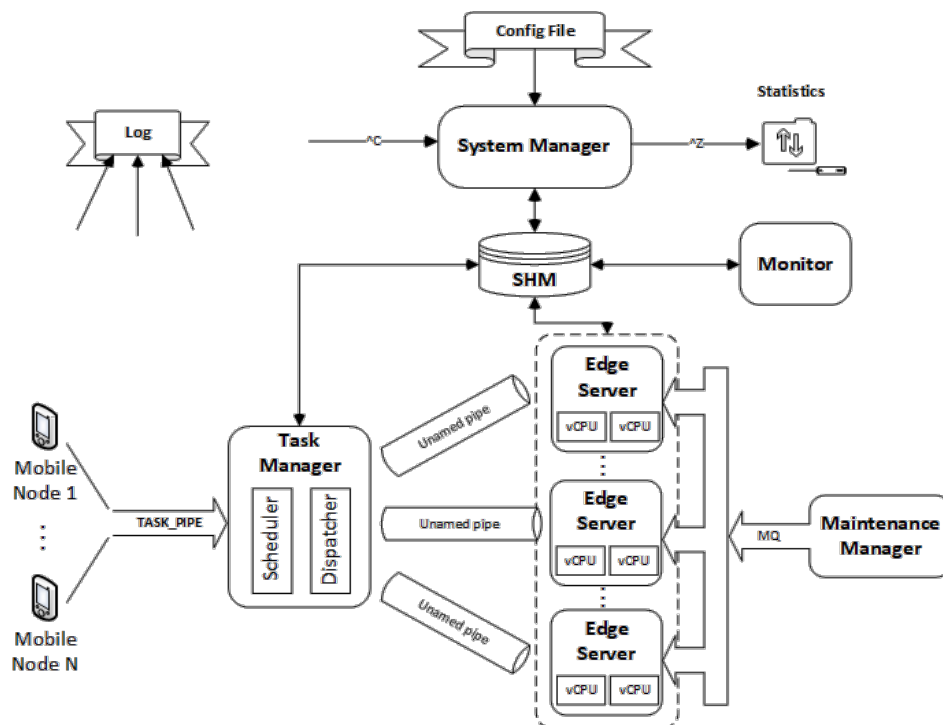


Figura 2. Visão geral do simulador a desenvolver.

Tal como é representado na Figura 2, o sistema é baseado em vários processos e *threads* que se descrevem a seguir:

- **Mobile Node** - Processo que gera tarefas para *offloading* e as envia pelo *named pipe*. Podem existir vários destes processos.
- **System Manager** - Processo responsável por iniciar o sistema, ler o ficheiro de configuração e criar os restantes processos envolvidos na execução de tarefas.
- **Task Manager** - Processo responsável por gerir a receção de tarefas e a sua distribuição pelos processos **Edge Server**.
- **Edge Server** - Processo responsável pela execução das tarefas. Dentro de cada **Edge Server** existem 2 vCPU (*virtual Central Processing Unit*) que são utilizados para executar as tarefas. Podem existir vários destes processos.
- **Monitor** - Processo responsável por controlar o número de vCPUs ativos dentro dos processos **Edge Server**.
- **Maintenance Manager** - Processo responsável pela manutenção dos servidores **Edge Server**.

E também por vários IPCs:

- **Named Pipe** - Permite enviar comandos e tarefas ao processo **Task Manager**.
- **SHM** - Zona de memória partilhada acedida pelos processos **System Manager**, **Task Manager**, **Edge Server** e **Monitor**. Deve conter todos os dados necessários à boa gestão do sistema.
- **Unnamed pipe** - Permitem a comunicação entre o processo **Task Manager** e cada um dos processos **Edge Server**.
- **MQ** - *Message Queue* que permite a troca de informação entre os **Edge Server** e o **Maintenance Manager**.

Existirá também um ficheiro de **log** onde serão escritas todas as informações 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 com detalhe as características e funcionalidades dos diversos componentes.

### ***Mobile Node***

Processo que gera tarefas com um determinado número de milhares de instruções (nº inteiro), a executar num intervalo de tempo definido. Estas tarefas são geradas um número pré-definido de vezes com um determinado intervalo. Cada um destes processos escreverá a tarefa no *named pipe* de nome **TASK\_PIPE**. Podemos ter um ou mais processos destes a correr em simultâneo, cada um com os seus parâmetros.

O número total de pedidos a gerar, o intervalo entre pedidos, o nº de instruções de cada pedido e o tempo máximo para executar cada um é fornecido por parâmetro de linha no arranque do processo.

Se quando tentar escrever no *named pipe*, este não existir, deve sair apresentando no ecrã esse erro.

Sintaxe do comando:

```
$ mobile_node {nº pedidos a gerar} {intervalo entre pedidos em ms}
{milhares de instruções de cada pedido} {tempo máximo para execução}
```

Informação a enviar para o *named pipe* relativa a cada tarefa:

ID tarefa; Nº de instruções (em milhares); Tempo máximo para execução

### ***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 o *named pipe* com o nome **TASK\_PIPE**, para que os dispositivos móveis possam fazer *offloading* de tarefas através do **Task Manager**.
- Cria o processo **Task Manager**
- Cria o processo **Monitor**
- Cria o processo **Maintenance Manager**
- Cria a fila de mensagens
- Cria a memória partilhada; a utilização da memória partilhada deve ser otimizada e não gastar espaço desnecessário
- Captura o sinal SIGTSTP e imprime estatísticas (ver conteúdo das estatísticas mais à frente neste enunciado)
- Captura o sinal SIGINT para terminar o programa.

Sintaxe do comando a executar na linha de comando:

```
$ offload_simulator {ficheiro de configuração}
```

### **Task Manager**

Processo responsável por gerir a recepção de tarefas através do *named pipe* e por as enviar para execução num dos processos **Edge Server**. Cria os processos **Edge Server** (cada processo representa um computador no Edge). Aceita comandos pelo *named pipe* com o nome **TASK\_PIPE**. Os comandos podem ser:

- Nova tarefa para *offloading*, enviada pelos processos **Mobile Node**
- Comando "EXIT" ou "STATS" que provocará respectivamente o fim do sistema de *offloading* ou a escrita de estatísticas
- Esses comandos serão escritos directamente pelo utilizador na linha de comando:
  - e.g., `echo "EXIT" > TASK_PIPE`

Quando uma nova tarefa de *offloading* chega através do *named pipe* são efetuadas as seguintes operações:

- O pedido é colocado numa fila com tamanho máximo de `QUEUE_POS` (este parâmetro é fornecido pelo ficheiro de configurações). Se a fila já estiver cheia, o pedido é eliminado e essa informação é escrita no ecrã e no ficheiro de **log**.
- Uma *thread scheduler* analisa as várias tarefas na fila e atribui-lhes prioridades (1= máxima prioridade). Sempre que um pedido novo chega e é colocado na fila, estas prioridades são reavaliadas. Se durante a avaliação forem detetadas tarefas cujo prazo máximo de execução já tenha passado, essas tarefas são eliminadas e essa informação será escrita no **log**. O critério para a prioridade é baseado no tempo máximo para execução que a tarefa tem e no seu tempo de chegada à fila. I.e., tarefas com limites temporais de execução menor têm prioridade. O tempo máximo para execução é o tempo máximo que a tarefa pode demorar até ser completamente executada, desde a altura em que chegou ao **Task Manager**.
- Uma *thread dispatcher* pega na tarefa mais prioritária e verifica se é possível executá-la no tempo disponível. I.e., verifica se o tempo que leva a executar em qualquer um dos vCPU disponível dos **Edge Server** cumpre o tempo máximo limite indicado - tem de ter em conta o tempo que ainda resta até ao limite temporal indicado e o tempo que levaria a processar no vCPU disponível. Se não cumprir o tempo limite, não vale a pena executar a tarefa, ela já não terá validade, pelo que a tarefa é eliminada e isso é escrito no **log**. A *thread dispatcher* é ativada sempre que 1 vCPU fica livre e desde que existam tarefas por realizar.

### **Edge Server**

- Cria até 2 *threads* para simular cada vCPUs.
- Mantém a memória partilhada atualizada com informação sobre a capacidade de processamento de cada um dos vCPUs em *million instruction per second* (MIPS), a altura em que cada vCPU poderá receber outra tarefa para executar e sobre o nível de performance do **Edge Server**.
- Os níveis de performance possíveis são:
  - *High performance*: o **Edge Server** tem os 2 vCPUs ativos
  - *Normal*: o **Edge Server** está em modo de economizar energia pelo que tem apenas 1 dos vCPUs ativo (o que tiver menos capacidade de processamento)
  - *Stopped*: o **Edge Server** está parado e os seus 2 vCPUs não estão a funcionar.
- Ao arrancar informa o **Maintenance Manager** que está ativo.
- Ao terminar uma tarefa escreve essa informação no **log**.

### **Monitor**

Processo responsável por controlar o número de vCPUs ativos dentro dos processos **Edge Server**. Para poupar energia cada servidor apenas tem por omissão 1 vCPU ativo.

Sempre que a fila no processo **Task Manager** esteja mais de 80% preenchida e o tempo mínimo de espera para que uma nova tarefa seja executada for superior a MAX\_WAIT segundos (ver ficheiro de configurações), o processo Monitor ativará o modo *High performance* de todos os **Edge Server**. Quando a fila descer a ocupação para 20% volta a ativar o modo de performance *Normal*. A troca de modo de performance será assinalada através de uma *flag* existente em memória partilhada.

### **Maintenance Manager**

De tempos a tempos é necessário fazer manutenção aos servidores **Edge Server**. Essa tarefa é gerida pelo processo **Maintenance Manager**. Quando é altura de manutenção de um dos **Edge Server**, o **Maintenance Manager** comunica com o servidor através da fila de mensagens, avisando-o que terá de entrar em manutenção. Ao receber essa mensagem o servidor termina primeiro as tarefas de que está a tratar e depois envia uma mensagem de volta ao **Maintenance Manager** a assinalar que está pronto para a intervenção. Durante esse tempo não poderá executar qualquer tarefa, passando para o modo *Stopped*.

Ao entrar em modo de manutenção, o **Edge Server** escreve essa informação no *log*.

Quando a manutenção estiver terminada o **Maintenance Manager** envia outra mensagem ao **Edge Server** a dizer-lhe que poderá continuar.

Nunca podem estar todos os servidores em manutenção simultânea. O intervalo de manutenção e o tempo de cada manutenção são aleatórios e variam entre 1 e 5s.

### **Atualização da Memória Partilhada**

Todos os processos que têm acesso à memória partilhada, devem mantê-la atualizada, usando os mecanismos necessários para que não seja possível a existência de corrupção de dados.

### **Fim controlado do simulador de offloading**

O sistema que controla e executa o *offloading* tem de terminar de forma controlada. Ao receber um SIGINT através do **System Manager** ou uma mensagem "EXIT" no **Task Manager**, o sistema segue as seguintes etapas:

- Escreve no log que o programa vai acabar.
- O **Task Manager** não recebe mais tarefas.
- Aguarda que todas as tarefas que estejam a executar nos **Edge Server** acabem.
- Escreve no log as tarefas que não chegaram a ser executadas.
- Após as tarefas terminarem, imprime as estatísticas da simulação e remove todos os recursos utilizados pelo sistema de *offloading* (contém todos os recursos usados, com exceção dos processos **Mobile Node**).

### **Estatísticas**

Ao receber um SIGTSTP através do **System Manager** ou uma mensagem "STATS" no **Task Manager**, o sistema escreve no ecrã as seguintes estatísticas:

- Número total de tarefas executadas
- Tempo médio de resposta a cada tarefa (tempo desde que a tarefa chegou até começar a ser executada)
- Número de tarefas executadas em cada **Edge Server**
- Número de operações de manutenção de cada **Edge Server**
- Número de tarefas que não chegaram a ser executadas

## Ficheiro de Configurações

O ficheiro de configurações deverá seguir a seguinte estrutura:

```
QUEUE_POS - número de slots na fila interna do Task Manager
MAX_WAIT - tempo máximo para que o processo Monitor eleve o nível de performance dos
Edge Servers (em segundos)
EDGE_SERVER_NUMBER - número de edge servers (>=2)
Nome do servidor de edge e capacidade de processamento de cada vCPU em MIPS
```

Exemplo do ficheiro de configurações:

```
50
2
3
SERVER_1,100,200
SERVER_2,150,200
SERVER_3,180,200
```

## Comandos a enviar através do *named pipe*

Tarefa: ID tarefa:Nº de instruções (em milhares):Tempo máximo para execução

Terminar sistema de *offloading*: EXIT

Escrever estatísticas no ecrã: STATS

## 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 **sempre** ser 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
- Erros ocorridos
- Mudança de estado de cada *Edge Server*
- Sinais recebidos

Exemplo do ficheiro de *log*:

```
18:00:05 OFFLOAD SIMULATOR STARTING
18:00:06 PROCESS TASK_MANAGER CREATED
18:00:06 PROCESS MONITOR CREATED
(...)
18:00:23 SERVER_1 READY
18:00:23 SERVER_2 READY
(...)
18:00:10 WRONG COMMAND => SAIR
(...)
18:02:00 SIGNAL SIGTSTP RECEIVED
(...)
18:04:00 DISPATCHER: TASK 100 SELECTED FOR EXECUTION ON SERVER_1
(...)
18:04:30 SERVER_1: TASK 100 COMPLETED
18:05:50 SIGNAL SIGINT RECEIVED
18:06:00 SIMULATOR WAITING FOR LAST TASKS TO FINISH
18:06:10 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?
<i>Mobile nodes</i>	Criação do mobile node	S
	Leitura correta dos parâmetros da linha de comando	S
	Geração e escrita das tarefas no <i>named pipe</i>	
<i>System Manager</i>	Arranque do simulador de <i>offloading</i> , leitura do ficheiro de configurações, validação dos dados do ficheiro e aplicação das configurações lidas.	S
	Criação da memória partilhada	S
	Criação do <i>named pipe</i>	
	Criação dos processos <i>Task Manager</i> , <i>Monitor</i> e <i>Maintenance Manager</i>	S
	Criação da fila de mensagens	
	Escrever a informação estatística no ecrã como resposta ao sinal SIGTSTP	
	Capturar o sinal SIGINT, terminar a corrida e liberta os recursos	
<i>Task Manager</i>	Criar os processos <i>Edge Server</i> de acordo com as configurações	S
	Ler e validar comandos lidos do <i>named pipe</i>	
	Criação da <i>thread scheduler</i> e gestão do escalonamento das tarefas	S (preliminar)
	Criação da <i>thread dispatcher</i> para distribuição das tarefas	
<i>Edge Server</i>	Criação das <i>threads</i> que simulam os vCPUs	S
	Executar as tarefas	
<i>Monitor</i>	Controla o nível de performance dos <i>Edge Server</i> de acordo com as regras estabelecidas	
<i>Maintenance Manager</i>	Gerar mensagens de manutenção, receber resposta e gerir a manutenção	
<i>Ficheiro log</i>	Envio sincronizado do <i>output</i> para ficheiro de <i>log</i> e ecrã.	S
Geral	Criar um <i>makefile</i>	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 <i>threads</i> que necessitem	
	Sincronização com mecanismos adequados (semáforos, <i>mutexes</i> ou variáveis de condição)	S (preliminar)
	Prevenção de interrupções indesejadas por sinais não especificados no enunciado; fornecer a resposta adequada aos vários sinais especificados no enunciado	
	Após receção de SIGINT, terminação controlada de todos os processos e <i>threads</i> , e libertação de todos os recursos.	

#### 4. Notas importantes

- **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.
- 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:
  - Esperas ativas serão fortemente penalizadas! Executem o comando `top` num terminal para se assegurarem 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 estratégias 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 não compilação do código enviado implica uma classificação de **ZERO valores** na meta correspondente.
- A defesa final do trabalho é obrigatória para todos os elementos do grupo. 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).
- A nota da defesa é individual pelo que cada um dos elementos do grupo poderá ter uma nota diferente;
- 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.
- Ambas as defesas, intermédia e final, devem ser realizadas na mesma turma e com o mesmo docente.



## 5. Metas, entregas e datas

Data	Meta	
<u>Data de entrega no Inforestudante</u> 04/04/2022 10h00 18/04/2022-22h00	Entrega intermédia	<p>Crie um arquivo no formato <b>ZIP (NÃO SERÃO ACEITES OUTROS FORMATOS)</b> com todos os ficheiros do trabalho e submeta-o no Inforestudante:</p> <ul style="list-style-type: none"> <li>• Os <b>nomes e números</b> dos alunos do grupo devem ser colocados <u>no início dos ficheiros com o código fonte</u>.</li> <li>• Inclua <u>todos</u> os ficheiros fonte e de configuração necessários e também um Makefile para compilação do programa.</li> <li>• <u>Não inclua</u> quaisquer ficheiros não necessários para a compilação ou execução do programa (ex. diretórios ou ficheiros de sistemas de controlo de versões)</li> <li>• Com o código deve ser entregue <b>1 página A4 com a arquitetura e todos os mecanismos de sincronização a implementar descritos</b>.</li> <li>• <b><u>Não serão admitidas entregas por e-mail.</u></b></li> </ul>
Semana de <u>04/04/2022</u> 18/04/2022	Demonstração /defesa intermédia	<ul style="list-style-type: none"> <li>• A demonstração deverá contemplar todos os pontos referidos na <i>checklist</i> que consta deste enunciado.</li> <li>• A demonstração/defesa será realizada nas aulas PL.</li> <li>• A defesa intermédia vale <b>20%</b> da cotação do projeto.</li> </ul>
<u>Data de entrega final no Inforestudante</u> 13/05/2022-22h00	Entrega final	<p>Crie um arquivo no formato <b>ZIP (NÃO SERÃO ACEITES OUTROS FORMATOS)</b> com todos os ficheiros do trabalho e submeta-o no Inforestudante:</p> <ul style="list-style-type: none"> <li>• Os <b>nomes e números</b> dos alunos do grupo devem ser colocados <u>no início dos ficheiros com o código fonte</u>.</li> <li>• Inclua <u>todos</u> os ficheiros fonte e de configuração necessários e também um Makefile para compilação do programa.</li> <li>• <u>Não inclua</u> quaisquer ficheiros não necessários para a compilação ou execução do programa (ex. diretórios ou ficheiros de sistemas de controlo de versões)</li> <li>• Com o código deve ser entregue um <b>relatório</b> sucinto (no máximo 2 páginas A4), no formato <b>pdf (NÃO SERÃO ACEITES OUTROS FORMATOS)</b>, 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.</li> <li>• <b><u>Não serão admitidas entregas por e-mail.</u></b></li> </ul>
16/05/2022 a 03/06/2022	Defesa final	<ul style="list-style-type: none"> <li>• A defesa final vale <b>80%</b> da cotação do projeto e consistirá numa análise detalhada do trabalho apresentado.</li> <li>• Defesas em grupo nas aulas PL.</li> <li>• É necessária inscrição para a defesa.</li> </ul>

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