Ciências ULisboa | Informática

Sistemas Operativos

2014/15

Miniprojeto 1

1 Introdução ao projeto

A parte teórico-prática da disciplina de Sistemas Operativos pretende familiarizar os alunos com alguns dos problemas envolvidos na utilização dos recursos de um Sistema Operativo. O projeto de avaliação será realizado utilizando principalmente a linguagem de programação C e as APIs de Linux e POSIX (*Portable Operating System Interface*), mas inclui também alguma programação de *Shell scripts* (`bash` e `makefile`).

O propósito geral do projeto é o desenvolvimento, de forma modular e faseada, de uma aplicação em C que criará múltiplos processos cooperativos que irão simular o fluxo central de um serviço de instalação de software e hardware em computadores (**SOinstala**). Este fluxo envolve o preenchimento de formulários e relatórios relacionados com o serviço deixando de fora, por simplificação, o fluxo do pagamento e emissão de fatura/recibo. De forma a se poder aferir a qualidade do serviço são registadas informações de progresso (log) que podem posteriormente ser analisadas.

O projeto SOinstala será realizado em 4 fases. A primeira fase do projeto tem como objetivo fundamental a criação de múltiplos processos cooperativos e das zonas de memória partilhada para comunicação entre si. A segunda fase foca-se na sincronização entre processos e, mais particularmente, na implementação do modelo produtor/consumidor que regula a utilização das zonas de memória partilhada. A terceira fase cobre a interação com ficheiros, a utilização dos mecanismos de medição de tempo e ativação de alarmes. Finalmente a quarta fase trata de aspetos envolventes à execução da aplicação por meio da criação de *Shell scripts* e melhoramento do `makefile`. Neste primeiro enunciado é feita uma apresentação geral do SOinstala juntamente com informação específica de suporte à realização da primeira fase. Em cada fase seguinte será disponibilizado um novo enunciado que complementará a informação contida nos anteriores.
Em resumo, as 4 fases de entrega têm os seguintes temas:

1. Miniprojecto 1 – Processos e memória
2. Miniprojecto 2 – Produtor/consumidor
3. Miniprojecto 3 – Ficheiros, tempo e relógios
4. Miniprojecto 4 – Shell script

2 Funcionamento geral

O SOinstala oferece um conjunto de serviços de instalação de software e hardware para os computadores dos clientes. Os serviços de reparação possíveis são: instalação de um sistema operativo, instalação de um pacote de segurança (com antivirus, antimalware e firewall) e instalação de vários tipos de hardware (CPU, memória, placa gráfica ...).
O fluxo central do sistema a simular, ver Figura 1, compreende as seguintes etapas:

1. Um cliente chega com o computador, consulta a lista de serviços da instalação, preenche o formulário de pedido de serviço indicando qual o serviço que pretende, coloca-o no cesto de pedidos de serviço e disponibiliza o seu computador;
2. Um rececionista obtém o próximo pedido de serviço do cesto respetivo, verifica a disponibilidade em stock do software, ou hardware, e se estiver disponível preenche um pedido de instalação que coloca no cesto de pedidos de instalação;

3. Um instalador obtém o próximo pedido de instalação do cesto respetivo, efetua a instalação solicitada, preenche o relatório de conclusão e coloca-o no cesto dos relatórios de conclusão;
4. Um cliente cujo serviço tenha sido concluído levanta o respetivo relatório de conclusão e o seu computador e sai.

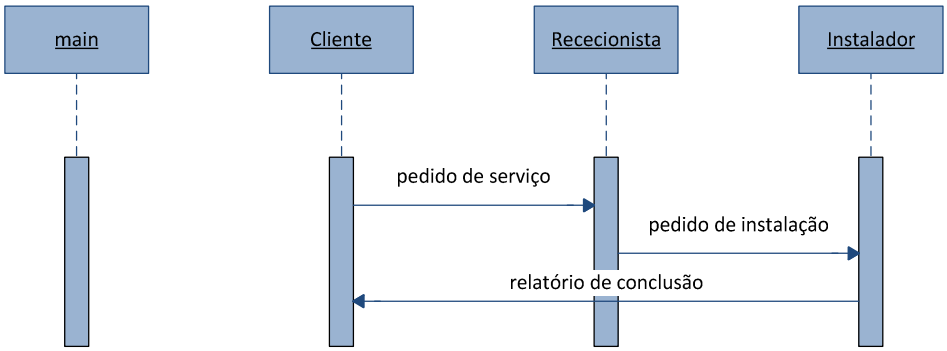

Figura 1 – Diagrama de sequência para a instalação de software/hardware

3 Interação com o Sistema Operativo

No SOinstala as pessoas são representadas por **processos** que cooperam através de buffers (cestos) para colocação e levantamento de pedidos/relatórios (Figura 2).

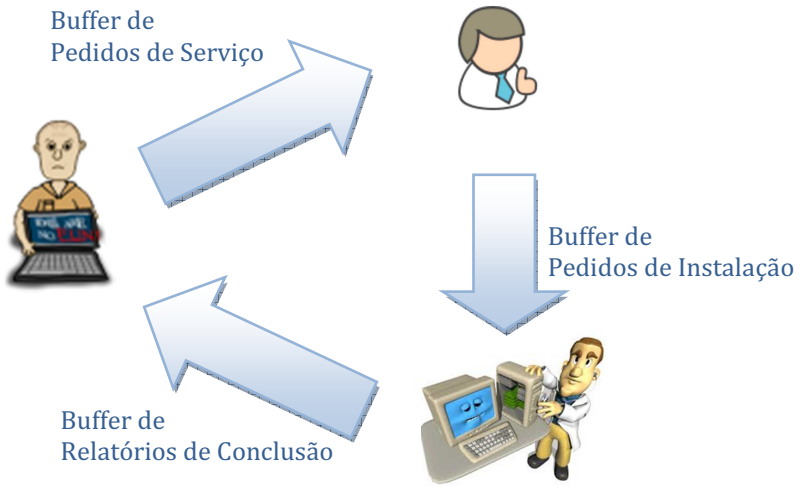

Figura 2 – Comunicação entre os processos através de memória partilhada

Os processos comunicam através da escrita e leitura em buffers que não são mais do que zonas de memória com uma determinada capacidade para guardar estruturas de dados de um dado tipo (vetor de estruturas). Cada estrutura de dados comporta um pedido, ou relatório, consoante o buffer.
Mas uma vez que um processo não pode aceder ao espaço de endereçamento de outro processo, estes buffers têm de ser concretizados através de **zonas de memória partilhada**. Os processos e as zonas de memória partilhada são explicados neste enunciado.
De modo a que o acesso às zonas de memória partilhada seja coerente, os processos terão de se **sincronizar**. Adicionalmente, os vários processos também necessitam de se **coordenar** na realização das suas atividades.
A configuração e a escrita de resultados é feita através de **ficheiros**. Desta forma a configuração pode ser feita sem necessidade de recompilação da aplicação e os resultados permanecem guardados após a execução do SOinstala.
A execução do SOinstala é acompanhada através da utilização de **relógios** que permitem registar o instante de cada estado que é recolhido. Os estados são armazenados em ficheiro para posterior análise. Adicionalmente através de **alarmes** é possível mostrar regularmente no ecrã informação sobre a evolução do sistema.
Finalmente, a organização e manutenção dos ficheiros do projeto é feita através de ***shell scripts***.

4 Configuração

A execução do SOinstala é parametrizada através de um ficheiro de configuração que descreve um determinado cenário. Em cada cenário são descritos precisamente quais os serviços disponibilizados, o nº de clientes e o serviço que solicitam, o nº de rececionistas, o nº de instaladores e os serviços que cada um cobre, e finalmente a capacidade dos diversos cestos de pedidos/relatórios (buffers). Esta abordagem permite testar facilmente o SOinstala em diferentes cenários.
Um cenário possível é o seguinte:

```
[SERVICOS]
; maximo de servicos = 10
; considera-se que os servicos são identificados sequencialmente por (0-9)
; o stock de cada servico e' indicado pela quantidade (0-...)
; neste caso são definidos os stocks de 6 servicos numerados de (0-5)
STOCK = 1 2 1 5 10 6

[CLIENTES]
; o n° de clientes e' dado pelo n° de elementos na lista
; os clientes nao tem identificadores
; o servico requisitado por cada um e' indicado pelo n° (0-9)
; neste caso e' definido o servico requisitado por cada um dos 31 clientes
SERVICO = 0 1 2 3 4 5 4 3 2 1 0 1 2 3 4 5 4 3 2 1 0 1 2 3 4 5 4 3 2 1 0

[RECECIONISTAS]
; o n° de rececionistas e' dado pelo n° de elementos na lista
; cada rececionista e' identificado por (R1, R2 ...)
; neste caso existem 2 rececionistas
LISTA = R1 R2

[INSTALADORES]
; o n° de instaladores e' dado pelo n° de elementos na lista
; os servicos que cada instalador sabe efetuar sao indicados pelos n° (0-9)
; as especialidades de cada instalador sao separadas por virgula
; neste caso existem 2 instaladores que sabem realizar qualquer instalacao
ESPECIALIDADES = 0 1 2 3 4 5,0 1 2 3 4 5

[BUFFERS]
; o n° de buffers não pode ser alterado
; cada buffer e' caracterizado por uma capacidade que indica o n° maximo de
; pedidos/relatorios que podem ser armazenados num dado instante
; os valores da capacidade podem ser alterados
CAPACIDADE = 2 2 10

; tamanho máximo da linha = 500 chars
```

5 Processos

Cada processo criado (para além do processo inicial) representará uma pessoa que poderá ter um dos papéis referidos, nomeadamente: cliente, rececionista ou instalador. Podem existir vários processos (pessoas) a desempenhar o mesmo papel.

O SOinstala encerra quando todos os clientes forem atendidos. Nessa situação todos os processos devem terminar e, de seguida, o processo inicial pode indicar a conclusão do serviço no SOinstala.

De seguida são especificadas as ações de cada tipo de processo através de diagramas de estado.

5.1 Principal

O processo principal (Figura 3) prepara o ambiente de comunicação e sincronização, abre o SOinstala, lança os processos que representam as várias pessoas e fica a aguardar pela sua conclusão. No final fecha o SOinstala, apresenta indicadores do funcionamento do sistema, liberta os recursos reservados para a comunicação e sincronização e termina.
Este processo realiza as instruções contidas no ficheiro `main.c`.

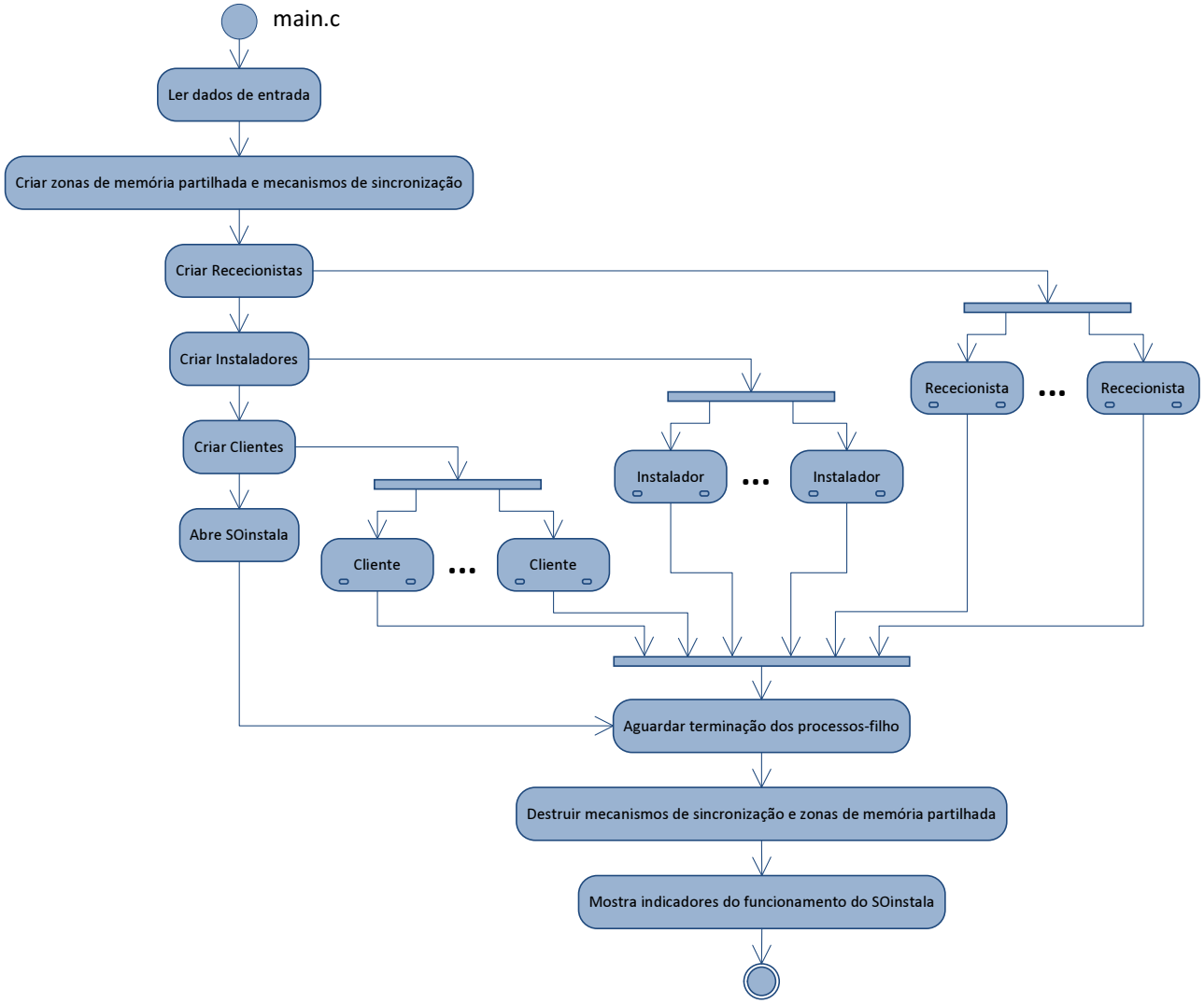

Figura 3 – Diagrama de estados do processo principal

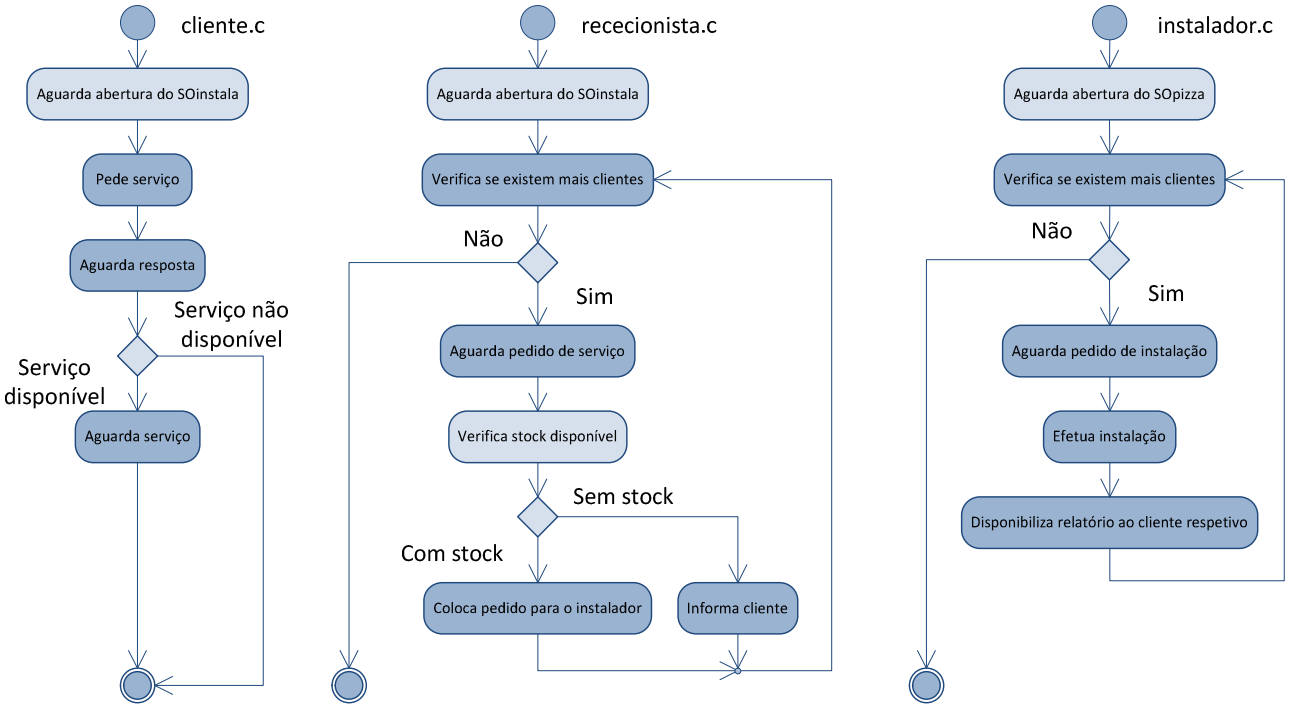

Figura 4 – Diagramas de estados dos processos cliente, rececionista e instalador

5.2 Cliente

Um processo cliente começa por aguardar a abertura do SOinstala e pede o serviço que está indicado no ficheiro de configuração. Se o serviço estiver disponível então aguarda a sua conclusão e abandona o SOinstala. Caso contrário abandona imediatamente o SOinstala. Ver Figura 4.

5.3 Rececionista

Um processo rececionista começa por aguardar a abertura do SOinstala e, enquanto existirem clientes, aguarda o pedido de serviço de um cliente. Ao chegar um pedido de serviço, se existir stock do componente respetivo então coloca um pedido de instalação a um instalador. Caso contrário informa o cliente de que o serviço não está disponível (sem stock). Ver Figura 4.

5.4 Instalador

Um processo instalador começa por aguardar a abertura do SOinstala e, enquanto existirem clientes, aguarda um pedido de um cliente. Ao chegar um pedido de instalação que ele possa tratar, prepara a respetiva instalação e, quando concluída informa o cliente respetivo. Ver Figura 4.

6 Estruturas de dados

De modo a permitir a comunicação entre os vários processos são estabelecidas várias estruturas de dados que agregam toda a informação. Estas estruturas de dados irão sendo apresentadas gradualmente em cada fase do miniprojeto.
Na 1ª fase deste miniprojeto são introduzidas as seguintes estruturas de dados:

- **servico**
- **configuracao**
- **indicadores**
- **ponteiros**
- **pedido_s**
- **pedido_i**
- **relatorio_c**

A estrutura `servico` representa: um pedido ao rececionista, um pedido ao instalador e também um relatório de instalação. Esta estrutura é criada pelo processo cliente como um pedido de serviço e passada ao rececionista. O rececionista recebe esta estrutura e no caso de existência de *stock* atualiza-a e passa-a ao instalador. O instalador recebe esta estrutura, efetua a instalação, atualiza a estrutura e entrega-a ao cliente respetivo. Os campos desta estrutura são:

```
struct servico {
    int id;                         // identificador do tipo de servico pedido (0, 1, …)
    int disponivel;                 // stock: 0 - indisponivel, 1 - disponivel
    int cliente;                    // id do cliente que encomendou o servico (0, 1, …)
    int rececionista;               // id do rececionista que atendeu o cliente (0, 1, …)
    int instalador;                 // id do instalador que efetuou o servico (0, 1, …)
    struct timespec hora_servico;   // hora a que o servico foi pedido ao rececionista
    struct timespec hora_instalacao; // hora a que a instalacao foi pedida ao instalador
    struct timespec hora_conclusao;  // hora a que o relatorio foi produzido pelo instalador
};
```

A estrutura `configuração` contém os dados lidos do ficheiro de configuração. Esta está organizada da seguinte forma:

```
struct configuracao {
    char* lista_servicos;    // linha com stock de cada servico
    char* lista_clientes;    // linha com servico pretendido por cada cliente
    char* lista_rececionistas;// linha com os rececionistas (nomes nao interessam)
    char* lista_instaladores;// linha com as especialidades por instalador separadas por virgula
    char* lista_buffers;     // linha com capacidade dos tres buffers
    int   SERVICOS;          // nº de servicos disponiveis
    int   CLIENTES;          // nº de clientes
    int   RECECIONISTAS;     // nº de rececionistas
    int   INSTALADORES;      // nº de instaladores
    int   BUFFER_SERVICO;    // capacidade do buffer de pedidos de servico
    int   BUFFER_INSTALACAO; // capacidade do buffer de pedidos de instalacao
    int   BUFFER_CONCLUSAO;  // capacidade do buffer de relatorios de conclusao
    int*  stock;             // vetor com capacidade por tipo de servico
};
```

A estrutura `indicadores` contém os dados que permitem no final da execução aferir a concretização dos objetivos. O *stock* inicial é aqui guardado de modo a se poder comparar no final com o stock restante que está na estrutura `configuracao`. Os campos são os seguintes:

```
struct indicadores {
    int *stock_inicial;
    int *pid_clientes;
    int *pid_rececionistas;
    int *pid_instaladores;
    int *clientes_atendidos_pelos_rececionistas;
    int *clientes_atendidos_pelos_instaladores;
    int *servicos_obtidos_pelos_clientes;
    int *servicos_realizados_pelos_instaladores;
};
```

A estrutura `ponteiros` dá suporte à utilização de um *buffer* circular. Para isso oferece dois índices: `in` e `out`. O índice `in` indica a próxima posição do *buffer* a ser escrita por um produtor. O índice `out` indica a próxima posição do *buffer* a ser lida por um consumidor.

A estrutura `pedido_s` contém um ponteiro para um *buffer* de serviços, onde cada *slot* pode conter uma estrutura `servico`, e um ponteiro para uma estrutura `ponteiros`, com os índices das próximas posições a serem lidas e escritas. Esta estrutura é utilizada para os clientes colocarem os seus pedidos de serviço para os rececionistas.

A estrutura `pedido_i` contém um ponteiro para um *buffer* de serviços, onde cada *slot* pode conter uma estrutura `servico`, e um ponteiro para um vetor de inteiros que indicam quais as posições do *buffer* que estão livres (valor é 0) ou ocupadas (valor é 1). Esta estrutura é utilizada para os rececionistas encaminharem os pedidos para os instaladores.

A estrutura `relatorio_c` contém um ponteiro para um *buffer* de serviços, onde cada *slot* pode conter uma estrutura `servico`, e um ponteiro para um vetor de inteiros que indicam quais as posições do *buffer* que estão livres (valor é 0) ou ocupadas (valor é 1). Esta estrutura é utilizada para os instaladores fazerem chegar os relatórios de conclusão aos clientes.

De realçar que cada *slot* das estruturas `pedido_s`, `pedido_i` e `relatorio_c`, contém sempre uma estrutura `servico`. Inicialmente o cliente cria uma estrutura `servico` e preenche alguns campos. Os restantes campos da estrutura devem ir sendo preenchidos nas etapas seguintes por um dos rececionistas e por um dos instaladores.

Estas estruturas de dados encontram-se definidas nos ficheiros .h do projeto.

NOTA IMPORTANTE: **O conteúdo dos ficheiros .h não pode ser alterado**.

7 Organização do projeto e geração do executável

Os ficheiros do projeto podem ser descarregados da página da disciplina. Para descomprimir o arquivo executar o seguinte comando:

```
$ tar -zxvf padrao.tar.gz
```

Depois de descomprimido o arquivo, serão encontrados os seguintes diretórios:

- `bin`
- `include`
- `logPlayer`
- `obj`
- `src`
- `testes`

O diretório `bin` deverá conter o executável `SOinstala`. O diretório `include` contém os ficheiros `.h` com a definição das estruturas de dados e declarações das funções. O diretório `logPlayer` contém o executável do depurador, `logPlayer`, que será apresentado na secção 9.3. O diretório `obj` contém os ficheiros objeto. O diretório `src` contém os ficheiros fonte que deverão ser alterados para realização do projeto. Finalmente o diretório testes inclui os vários ficheiros de configuração, ou cenários, do SOinstala e é também onde devem ser guardados os ficheiros gerados.

O executável do SOinstala pode ser gerado executando o `makefile`. Para isso basta dar o seguinte comando no diretório raiz do projeto:

```
$ make
```

8 Objetivos do miniprojeto 1

Este miniprojeto inclui dois conjuntos de objetivos relacionados com: processos e alocação de memória. Para a concretização de cada objetivo devem ser alteradas as zonas de código rodeadas com comentários do tipo "// ==========". Para cada zona devem ser comentadas quaisquer chamadas a funções que aí se encontrem e escrito novo código. O nome das funções já presentes nessas zonas é indicativo daquilo que deve ser concretizado. Desta forma cada zona pode ser desenvolvida e testada de forma independente.
Os vários objetivos são apresentados nas secções seguintes.

8.1 Processos

Esta primeira fase envolve a escrita de código no ficheiro `main.c`.
Os objetivos específicos desta fase são os seguintes:

1. Adicionar o código para tratar os parâmetros de entrada tal como especificados no código fonte.
2. Adicionar o código que permita criar todos os processos filho (clientes, rececionistas e instaladores).
3. Adicionar o código que permita ao processo principal aguardar a terminação dos filhos e registar o estado que cada um devolveu, através da chamada de sistema `exit`, na estrutura de dados **indicadores**.

8.2 Alocação de memória

Esta fase envolve a escrita de código no ficheiro `memoria.c`.
Os objetivos específicos desta fase são os seguintes:

4. `memoria.c`: Criação e terminação de um ***buffer* circular** (`BServico`), como zona de memória partilhada entre processos independentes, para os clientes colocarem as encomendas para os rececionistas. O tamanho do *buffer* deverá estar de acordo com o valor contido em `Config.BUFFER_SERVICO` (estrutura `configuracao`).
5. `memoria.c`: Criação e terminação de um ***buffer* de acesso aleatório** (`BInstalacao`), como zona de memória partilhada entre processos independentes, para os rececionistas colocarem os pedidos para os instaladores. O tamanho do *buffer* deverá estar de acordo com o valor contido em `Config.BUFFER_INSTALACAO` (estrutura `configuracao`).
6. `memoria.c`: Criação e terminação de um ***buffer* de acesso aleatório** (`BConclusao`), como zona de memória partilhada entre processos independentes, para os instaladores colocarem os relatórios de conclusão para os clientes. O tamanho do *buffer* deverá estar de acordo com o valor contido em `Config.BUFFER_CONCLUSAO` (estrutura `configuracao`).
7. `memoria.c`: Adicionar o código que permite aos clientes, rececionistas e instaladores escreverem e lerem a informação dos *buffers* partilhados. Funções a alterar: `memoria_pedido_s_escreve`, `memoria_pedido_s_le`, `memoria_pedido_i_escreve`, `memoria_pedido_i_le`, `memoria_relatorio_c_escreve` e `memoria_relatorio_c_le`.
8. `memoria.c`: Adicionar o código que permita actualizar os campos da estrutura `servico` nas etapas adequadas, ex: na função `memoria_relatorio_c_escreve` o rececionista deve colocar o seu `id` no campo `rececionista`.
9. `memoria.c`: Na função `memoria_criar_indicadores`, reservar e libertar memória dinamicamente para cada um dos campos da estrutura indicadores com os tamanhos necessários (consultar estrutura `configuracao`). Iniciar o vetor `stock_inicial` com os valores contidos em `Config.stock` e apagar o conteúdo dos vetores `clientes_atendidos_pelos_rececionistas`, `clientes_atendidos_pelos_instaladores`, `servicos_obtidos_pelos_clientes` e `servicos_realizados_pelos_instaladores`.

O nome a atribuir a cada zona de memória partilhada não está estabelecido, devendo ser escolhido pelo grupo utilizando uma nomenclatura consistente.

9 Teste dos objetivos

Os métodos de teste passam pela análise do ficheiro de configuração, dos ficheiros gerados (resultados e log) e nalguns casos também das saídas para a consola (`stdout`). A concretização de cada um dos objetivos desta fase do miniprojeto pode ser verificada de acordo com o método indicado nas subsecções seguintes.

9.1 Processos

Os testes são para cada objetivo, respetivamente, os seguintes:

1. Verificar que o `SOinstala` reage à introdução de diferentes combinações de parâmetros.
2. Para um nº significativo de clientes, verificar que todos os clientes foram atendidos e que a ordem de atendimento não será sequencial.
3. Verificar que todos os clientes foram atendidos antes do processo principal dar o `SOinstala` como fechado e que os **indicadores** mostram informação válida.

9.2 Alocação de memória

Os testes para esta fase do miniprojeto são muito simples. Basta no final constatar que cada cliente obtém o serviço que pediu. O ficheiro de log permitirá aferir especificamente o funcionamento de cada *buffer* (ver secção 9.3). Finalmente os indicadores deverão produzir informação válida.

9.3 Depuração com o utilitário `logPlayer`

O `logPlayer` permite ver o conteúdo (ou só algumas partes) de um ficheiro de log que pode ser gerado pelo `SOinstala`. Este ficheiro de log pode ser muito útil para fazer depuração do programa por mostrar o conteúdo dos 3 *buffers* de comunicação cliente/rececionista/instalador sempre que são alterados.

Para gerar um ficheiro de log com o `SOinstala` utilizar a opção "`-l`" seguida do nome do ficheiro de log a criar.

O ficheiro de log é constituído por registos separados por uma sequência de 4 inteiros com todos os bits a 1. Cada registo contém o tempo decorrido (`double`), a etapa em que foi gerado (`int`), o id de quem o gerou (`int`): cliente, rececionista ou instalador; e o conteúdo das posições ocupadas dos 3 *buffers*, respetivamente: `BServico`, `BInstalacao` e `BConclusao`; (múltiplos de 4 `int`).

A etapa em que foi gerado é um número natural (1, 2 ...) cujo valor corresponde, respetivamente às seguintes etapas no acesso aos *buffers*:

1. Cliente escreve pedido de serviço
2. Rececionista lê pedido de serviço
3. Rececionista escreve pedido de instalação
4. Instalador lê pedido de instalação
5. Instalador escreve relatório de conclusão
6. Cliente lê relatório de conclusão

O conteúdo de cada *buffer* está separado por uma sequência de 2 inteiros com todos os bits a 1 e mais 2 inteiros com todos os bits a 0.

O conteúdo de cada *buffer* é constituído por 0, ou mais, entradas de 4 `int` cada. Cada entrada contém 4 identificadores que são: o id do serviço (`int`), o id do cliente (`int`), o id do rececionista (`int`) e o id do instalador (`int`).

O utilitário pode ser chamado passando apenas o nome do ficheiro de log a mostrar. Mas podem também ser especificados, na linha de comando, os seguintes filtros: etapa a visualizar e/ou id do cliente/rececionista/instalador a visualizar e/ou o nº do primeiro registo a mostrar e/ou o nº do último registo a mostrar. Considera-se que os registos a visualizar são numerados sequencialmente a partir de 1, em dependência dos filtros escolhidos.

Para conhecer as opções disponíveis basta executar o `logPlayer` sem parâmetros.

10 Entrega

Os ficheiros `main.c` e `memoria.c` devem ser entregues até às **20h** do dia **7 de Abril de 2015 (terça-feira)**, colocando-os no diretório **TRAB1** (maiúsculas) na respetiva área de grupos.

Não serão aceites trabalhos por e-mail nem por qualquer outro meio não definido nesta secção. Se não se verificar algum destes requisitos o trabalho é considerado como não entregue.