Sistemas Operativos 18/19

Trabalho Prático - Programação em C para UNIX

O trabalho prático de sistemas operativos consiste na implementação de um editor de texto multiutilizador aqui denominado `MEDIT`. O editor exibirá uma interface em modo consola (modo texto), e não envolverá questões de formatação de texto (justificação, alinhamento, etc. – nada disso está envolvido aqui). A complexidade do trabalho estará essencialmente na parte relacionada com os mecanismos Unix.

O editor é composto por uma aplicação cliente, invocada por cada um dos utilizadores, e uma aplicação servidor, lançada pelo administrador do sistema, responsável por coordenar o processo de edição. O servidor e os vários clientes residem na mesma máquina Unix. A troca de informação entre os clientes e o servidor é assegurada exclusivamente por mecanismos de comunicação inter-processo dados nas aulas. Este trabalho não envolve mecanismos de comunicação em rede. A utilização por vários utilizadores envolverá apenas consolas (terminais) diferentes, uma por utilizador. Dado que uma máquina dispõe apenas de um teclado e um ecrã, a utilização de sessões remotas (recorrendo ao *PuTTY*, por exemplo) simplificará bastante a utilização e teste em cenário de multi-utilizador, para a mesma máquina, na qual residirão as aplicações cliente e servidor que irá desenvolver.

São descritas primeiro as funcionalidades e no final as regras de funcionamento do sistema de edição.

Importante: existem diversas referências a "opcional" neste enunciado. Não se trata de funcionalidade cuja elaboração possa não ser feita. Trata-se apenas de argumentos da linha de comando do servidor e do cliente que podem, ou não, ser especificados no lançamento destas aplicações. Não o sendo entram em vigor valores por omissão mencionados neste enunciado. Em suma, todos os argumentos de linha de comando rotulados de "opcionais" devem ser implementados mas podem ou não ser especificados no lançamento das aplicações.

1. Descrição geral

O trabalho prático consiste no desenvolvimento de um editor de texto em modo consola para vários utilizadores. Cada utilizador é identificado por um *username* único solicitado no início do programa cliente ou passado na linha de comando (argumento opcional do cliente: **`-u`** *`username`*) devendo o cliente suportar ambas as modalidades. Não deve ser desenvolvido nenhum processo de autenticação de utilizadores (através de *password*, por exemplo). Apenas se realiza um processo de identificação através de *username*. Este *username* é específico do `MEDIT`, e em nada se relaciona com as contas de utilizadores do sistema operativo.

Para tanto deverá existir no servidor uma base de dados de *usernames*. Por omissão a base de dados consiste num ficheiro de texto `medit.db` presente na diretoria em que é lançado o servidor. No arranque do servidor pode ser especificada uma base de dados alternativa de *usernames* (argumento opcional do servidor: **`-f`** *`filename`*). O servidor deve portanto suportar ambas as formas (implícita e explícita) de localizar a base de dados.

A base de dados de *usernames* é um ficheiro de texto no qual cada linha é composta por um *username* de 8 caracteres no máximo (os caracteres excedentes são simplesmente ignorados). Não existe *password* nenhuma.
A arquitectura do sistema segue o paradigma cliente-servidor. O servidor controla o fluxo de informação entre os clientes e gere todas as funcionalidades do editor. Os clientes funcionam apenas como interface com os utilizadores. As funcionalidades principais são as descritas abaixo.

Formato do texto

O texto a ser editado deverá caber num único ecrã, ou seja, consistir numa matriz composta no máximo, por omissão, por 15 linhas de 45 colunas. Deve ser possível especificar um número de linhas e/ou colunas inferior no arranque do servidor através das variáveis de ambiente `MEDIT_MAXLINES` e `MEDIT_MAXCOLUMNS`. Se forem definidas substituirão os valores 15, 45 ou ambos, atrás mencionados. A unidade de edição do texto é a linha do ecrã (e não o parágrafo, como acontece nos editores comuns).

Veridficação ortográfica

O servidor verificará o conteúdo da nova linha quando o utilizador dá por terminada a edição dessa linha. A verificação consiste em saber se a palavra existe ou não. A verificaçãp é feita com recurso ao dicionário do pacote GNU Aspell. A verificação é feita usando *pipes* entre o servidor e o aspell. O servidor deverá lançar o dicionário e mantê-lo em execução. Para cada palavra que deseja verificar, envia-lhe essa palavra e recolhe o resultado usando *pipes*. No site abaixo encontram-se os pormenores acerca do funcionamento do aspell.

http://aspell.net/man-html/Through-A-Pipe.html#Through-A-Pipe

Mais abaixo são dados mais pormenores acerca de quando é que o servidor verifica as palavras e o que acontece quando encontra palavras não reconhecidas.

Utilização em paralelo

O texto pode ser editado por vários utilizadores em simultâneo. Todas as alterações são de imediato visíveis a todos os utilizadores. Cada utilizador trabalha numa linha de cada vez e a mesma linha não pode estar a ser editada em simultâneo por mais que um utilizador. Os utilizadores podem movimentar-se pelo texto livremente usando as teclas de cursor. Para editar determinada linha um utilizador deve premir a tecla `ENTER` enquanto estiver sobre a linha. Se essa linha já estiver a ser editada por outro utilizador, nada acontecerá. Se a linha estiver "livre", então fica "na posse" desse utilizador que a poderá então editar. É sempre o servidor que coordena (centralmente) os pedidos de edição de cada linha. Para terminar a edição de uma linha, libertando-

a, o utilizador deve premir novamente a tecla ENTER. Se o utilizador pretender cancelar todas as alterações que já efetuou na linha que está a editar deve premir a tecla ESC.

No ecrã cada linha deve ser precedida do seu número (00, 01, ...) e do *username* responsável pela edição da mesma se esta se encontrar a ser editada. As linhas que ainda não tenham sido definidas devem surgir de forma vazia e apenas precedidas do seu número.

Edição de texto

O utilizador estará apenas num de dois modos: *Modo de navegação no texto*, *Modo de edição de linha*.

- ***Modo de navegação no texto***. Neste modo o utilizador pode:
  - Movimentar-se ao longo do texto com as teclas de cursor.
  - Na movimentação as frases por definir podem ser tratadas como preenchidas na sua totalidade por espaços em branco.
  - Entrar em *Modo de edição de linha* (através da tecla ENTER) .
- ***Modo de edição de linha***. Durante a edição da linha o utilizador que está a editar a linha pode:
  - Deslocar-se para a esquerda ou direita nessa linha com as teclas de cursor, mas não pode deslocar-se para outra linha (apenas quando terminar a edição).
  - Inserir caracteres na posição actual do cursor. Os caracteres que estiverem mais para a direita são "empurrados" para a direita desde que ainda haja espaço. Se não houver espaço não são aceites mais caracteres (espaços à direita são considerados caracteres vazios).
  - Apagar caracteres, deslocando ("puxando") caracteres que estiverem mais à direita para a esquerda.
  - Terminar o *Modo de edição de linha*, submetendo as alterações introduzidas e voltando ao *Modo de navegação no texto* através da tecla ENTER. É neste momento que o servidor valida as palavras na linha. Se o servidor as aceitar como válidas o utilizador regressa ao Modo de Navegação. Caso contrário o utilizador será de alguma forma alertado para o problema encontrado e permanecerá no modo de edição para que o possa corrigir. A informação acerca de palavras não reconhecidas é visível apenas para o cliente em questão.
  - Terminar o *Modo de edição de linha*, descartando as alterações introduzidas e voltando ao *Modo de navegação no texto* através da tecla ESC. O servidor restaurará a linha com o conteúdo existente no instante anterior ao início da sua edição.
  - Todos os caracteres introduzidos durante o modo de edição são imediatamente visíveis a todos os utilizadores.

Aspectos de arquitectura do sistema

- Cliente/servidor com *named pipes*

  O servidor tem 1+N *named pipes*. Existe 1 *named pipe* principal e N *named pipes* de interacção. O *named pipe* principal destina-se a receber pedidos de ligação de novos utilizadores; os restantes são usados para atender clientes entretanto recebidos. Os clientes dirigem-se ao *named pipe* principal

para solicitar uma ligação ao sistema de edição, recebendo de volta (se positivamente identificados na base de dados de *usernames* registados) a indicação de qual o *named pipe* de interacção que devem usar daí em diante. O servidor atribui ao novo utilizador o *named pipe* de interacção que está a servir menos clientes nesse instante. O argumento (opcional) de linha de comando no servidor e no cliente **-p** *main-named-pipe* permite substituir o nome por omissão usado para o *named pipe* principal. O argumento (opcional) de linha de comando no servidor **-n** *number-of-interactive-named-pipes* permite substituir no servidor o valor por omissão do número de *named pipes* de interacção.

Outras características de funcionalidade

- Estatísticas

  O servidor fornece estatísticas relativas ao texto em edição, contabilizando o número total de palavras, o número de letras e os 5 carateres mais comuns (por ordem decrescente). Essa informação reflecte também as linhas que estão em edição, é continuamente actualizada ao segundo, e é comunicada a todos os clientes. Os clientes devem manter os respetivos utilizadores atualizados sobre as estatísticas assim que as recebam.

- *Timeout* por inactividade

  Uma linha em edição que não tenha alterações durante `10` segundos é considerada livre para edição por qualquer utilizador. O valor por omissão de `10` pode ser redefinido no servidor através da variável de ambiente `MEDIT_TIMEOUT`. A libertação de uma linha por *timeout* tem os mesmos efeitos que a libertação voluntária através da tecla `ENTERESC` (conteúdo novo abandonado).

- Interface visual e percepção visual da edição

  Todas as modificações de texto são sempre visíveis a todos os clientes. Isto inclui o estado de em-edição/livre das linhas. Esta visualização pode incluir mudanças de cor do texto ou do fundo. Os detalhes exactos ficam em aberto.

  A interface visual do cliente deve envolver a utilização da biblioteca *ncurses.* A biblioteca *ncurses* é contém diversas funções para impressão de caracteres, leitura de informação via teclado. Posicionamento dew cursor e definição de cores. O seu uso é feito em moldes semelhantes a qualquer outra biblioteca: "*existem estas funções, têm estes parâmetros, invocam-se desta forma*". Serão dadas algumas indicações nas aulas. Entre outros, consultar o site:

  http://tldp.org/HOWTO/NCURSES-Programming-HOWTO/)

- Administração do servidor

  O servidor recebe do administrador (utilizador que o lançou) comandos escritos. Estes comandos são aceites e processados de forma concorrente (em paralelo) com as demais tarefas do servidor. As operações de administração são essencialmente gestão de utilizadores e carregamento e salvaguarda do texto, abaixo detalhadas.

2. Arquitectura e detalhes adicionais de funcionalidade

O sistema é obrigatoriamente constituído por vários processos a correr na mesma máquina UNIX, processos esses que interagem entre si através de uma arquitectura cliente/servidor. Estes processos executam um de dois programas: "servidor" ou "cliente". O administrador do `MEDIT` lança o servidor e os utilizadores que pretendem colaborar na criação do texto lançam o respectivo cliente.

Servidor

Este programa armazena e gere toda a informação relativa à edição do texto, controlando todos os aspectos desta. Só pode estar a correr um processo com este programa de cada vez.
Este programa não é destinado ao utilizador que pretende editar o texto, mas sim a um utilizador administrador do `MEDIT`, o qual configura/controla o ambiente de edição, permitindo executar acções administrativas como o armazenamento/carregamento de dados do disco, gestão de utilizadores e visualização do estado actual do servidor. A interacção com o utilizador é feita através do paradigma de linha de comandos.

O servidor pode ser controlado directamente através de uma interface semelhante a uma linha de comandos (comandos escritos com parâmetros). Os comandos que o servidor reconhece são os seguintes:

- **`settings`**
  Apresenta os parâmetros de funcionamento actuais do servidor (número de linhas, número de colunas, nome completo da base de dados de *usernames*, número máximo de utilizadores, número de *named pipe* de interacção, nome do *named pipe* principal, ...).
- **`load`** `<filename>`
  Carrega o número máximo de linhas possível do ficheiro de texto especificado para edição. O conteúdo anterior do editor é descartado e todas as linhas deixam de estar em edição. As linhas do ficheiro com vocábulos não reconhecidos pelo dicionário ou extensão superior à máxima permitida são descartadas.
- **`save`** `<filename>`
  Salvaguarda no ficheiro mencionado o conteúdo actual do editor (validado ortograficamente). Todas as linhas são compulsivamente libertadas e os vocábulos são verificados antes da gravação.
- **`free`** `<linenumber>`
  Liberta compulsivamente a linha especificada. Descarta eventuais alterações entretanto introduzidas.
- **`statistics`**
  Mostra as estatísticas coligidas de um em um segundo (enquanto não é introduzido outro comando, mesmo que seja um comando inválido).
- **`users`**
  Mostra os utilizadores presentes ordenados por idade da respetiva sessão. Para cada *username* apresenta a idade da sessão, o nome do *named pipe* de interacção, a percentagem de linhas de sua autoria no texto actual e a linha que se encontra a editar nesse instante (se for o caso).
- **`text`**

Mostra o texto presente no mesmo formato que é apresentado aos clientes (número de linha e *username* que a está a editar, se for o caso, seguido da linha propriamente dita).

- **`shutdown`**
  Termina a edição em curso sem gravar qualquer alteração e informando previamente os clientes activos.

Cliente

Este programa serve de interface com os utilizadores que pretendem editar o texto, sendo responsável pela interacção com estes e por apresentar a versão actual do texto. Sendo a sua principal missão interagir com os utilizadores, este programa **não valida quaisquer regras de funcionamento do `MEDIT` nem armazena informação localmente** acerca do estado actual do texto. Cada utilizador executa um e só um cliente (validação feita no servidor). Aquando da execução, o utilizador identifica o seu *username* e, se este for validado positivamente pelo servidor, junta-se à lista de utilizadores que podem editar o texto.
Todos os clientes apresentam uma visão actualizada do texto, independentemente de o utilizador estar a editar uma linha do texto ou estar simplesmente a apreciar o que se passa.

Interface com o utilizador

Os requisitos da interface com o utilizador são, de uma forma geral, os seguintes: deve ser fácil de utilizar, deve fazer sentido e deve permitir editar uma linha do texto. As teclas utilizadas são as seguintes:

- `UP`, `DOWN`, `LEFT` e `RIGTH` – Movimentar-se ao longo do texto no *Modo de navegação no texto*, ou apenas da linha no *Modo de edição de linha*.
- `ENTER` – Comutar entre os dois modos de funcionamento (*Modo de navegação no texto* e *Modo de edição de linha*).
- `ESC` – Abandonar o *Modo de edição de linha* descartando todas a alterações efectuadas à linha de texto.
- `DEL` e `BACKSPACE` – Apagar caracteres, deslocando os caracteres que estiverem à direita para a esquerda.

O processo servidor interage com os processos cliente apenas através de *named pipes* e *sinais*. Os processos cliente não interagem diretamente uns com os outros. Qualquer informação que se queira transferir de um cliente para outro terá sempre de passar pelo servidor, sendo que este mediará e reencaminhará essa informação para o cliente destino.

Todos os programas são lançados através da linha de comandos. Cada programa deverá ser lançado a partir de uma sessão/*shell* diferente (localmente ou através do *PuTTY* para utilizadores remotos), mas utilizando a mesma conta de utilizador do sistema operativo subjacente.

Existe um número máximo de utilizadores que podem estar activos em simultâneo. Este número é definido pela variável de ambiente `MEDIT_MAXUSERS`. Pode assumir que esta variável terá sempre um valor entre 1 e `MEDIT_MAXLINES` (ou 15 se esta variável de ambiente não se encontrar definida) sendo o seu valor por omissão 3.

3. Requisitos e descrição adicionais

Os requisitos aqui apresentados definem aspectos adicionais que deve obrigatoriamente cumprir. Servem também para entender a lógica geral e deduzir algum aspecto omisso. SAlguma informação aqui apresentada já foi indicada antes, mas pode usar esta repetição para incorporar melhor os requisitos. É importante verificar com cuidado toda a descrição dada atrás.

Servidor

- Apenas deve existir um processo servidor em execução. Caso seja executada uma nova instância do servidor, esta deverá detetar a existência da primeira e terminar de imediato.
- Quando termina, o servidor deve informar todos os clientes do sucedido.
- O servidor deverá terminar caso receba o sinal SIGUSR1 ou comando de administração `shutdown`.
- O acesso aos dados por várias *threads* deve ser acautelado através de mecanismo de sincronização (*mutexes* ou semáforos).
- O servidor poderá ter que dar atenção a mais do que uma entrada de dados em simultâneo (por exemplo, *pipes* e teclado). Esta situação deve ser resolvida de forma elegante usando os mecanismos estudados (*threads*, *select*, etc.).

Cliente

- Podem estar a correr vários processos cliente em simultâneo (um por utilizador editor).
- A qualquer instante o utilizador poderá abandonar a edição do texto ou mesmo abandonar o programa de todo. O servidor deverá ser informado desta situação e agir em conformidade.
- A interface do cliente é totalmente em "modo-texto". **A construção da interface do cliente será feita com recurso à biblioteca *ncurses*.**
- O cliente manterá uma visão do texto permanentemente actualizada: mesmo que o utilizador desse cliente não faça nada (não se encontre a editar qualquer linha). Os outros utilizadores poderão estar a editar o texto, e portanto o cliente manterá o utilizador sempre actualizado relativamente ao conteúdo do mesmo. Este aspeto não é complexo, mas exige algum planeamento inicial quanto à lógica de comunicação (ver também o próximo ponto).
- O cliente poderá ter que dar atenção a mais do que um assunto (em particular, a várias entradas de dados) em simultâneo (por exemplo, informações do utilizador, teclado, mensagens do servidor, etc.). Esta situação deve ser resolvida de forma elegante usando os mecanismos estudados (*threads*, *select*, etc.).

4. Considerações adicionais

Os alunos devem ter em conta as seguintes considerações finais:

- Podem existir diversas situações e potenciais erros que não são explicitamente mencionados no enunciado. Estas situações devem ser identificadas e os programas devem lidar com elas de uma forma controlada e estável. Um programa terminar abruptamente perante uma destas situações não é considerada uma forma adequada.
- Caso existam *bugs* no enunciado estes serão corrigidos e a sua correcção será anunciada. Poderá haver lugar a documentos adicionais no *moodle*, nomeadamente um *Frequently Asked Questions* com as perguntas mais frequentes e respectivas respostas.

5. Regras gerais do trabalho, METAS e DATAS IMPORTANTES

Aplicam-se as seguintes regras, descritas na primeira aula e na ficha da unidade curricular (FUC):

- O trabalho pode ser realizado em grupos de um ou de dois alunos (sem excepções, e por isso não vale a pena os alunos enviarem e-mails com pedidos nesse sentido). No caso de ser realizado individualmente, o trabalho (e a valorização) é o mesmo.
- Existem defesas obrigatórias, também descritas na FUC. A defesa será efetuada em moldes a definir via *moodle* na altura em que tal for relevante.
- Existem três metas ao todo, tal como descrito na FUC. As datas e requisitos das metas são indicados mais abaixo. Em todas as metas a entrega é feita via *moodle* através da submissão de um único **arquivo *zip***[1] cujo **nome** segue a seguinte lógica[2]:

  so_1819_tp_meta1_*nome1_numero1_nome2_numero2*.zip

  (evidentemente, meta1, nome e número serão adaptados à m eta, nomes e números dos elementos do grupo)

  A não aderência ao formato *zip* e ao formato do nome do ficheiro será penalizada.

Metas: requisitos e datas de entrega

Meta 1:

Requisitos:

- Planear e definir as estruturas de dados responsáveis por gerir as definições de funcionamento no servidor e no cliente. Definir *header files* (exemplo: `medit-defaults.h`, `client-defaults.h` e `server-defaults.h`) com constantes simbólicas (`#define` ...) que registem os valores por omissão comuns e específicos do cliente e servidor.

[1] Leia-se "***zip***" - não é *arj*, *rar*, *tar*, ou outros. O uso de outro formato será **penalizado**. Há muitos utilitários da linha de comando UNIX para lidar com estes ficheiros (zip, gzip, etc.). Use um.
[2] O não cumprimento do formato do nome será **penalizado**.

- Desenvolver a lógica de leitura das variáveis de ambiente do servidor e do cliente, reflectindo-se nas estruturas de dados mencionadas no ponto anterior. Sugestão: usar as funções `getopt()` e `getenv()`.
- Iniciar o desenvolvimento da consola de administração do servidor implementando a leitura e validação dos comandos e respectivos parâmetros e a implementação completa do comando `settings`.
- Definir uma estrutura de dados responsável por acolher o texto em edição e implementação da edição de uma linha com as funcionalidades descritas de inserção de caracteres, eliminação de caracteres e navegação dentro dessa linha. Nesta funcionalidade já faz sentido o uso da biblioteca ncurses. Nesta funcionalidade não existem ainda vários utilizadores e não é necessário considerar a questão dos vocábulos.
- Desenvolver uma função/mecanismo que dado um username e o nome do ficheiro de usernames verifica se este se encontra nesse ficheiro.

Data de entrega: Domingo, 28 de Outubro. Sem possibilidade de entrega atrasada.

Meta 2:

Requisitos:

- Todos os requisitos da Meta 1.
- Comunicação através de *named pipes* a funcionar (servidor-cliente)-
- Gestão dos clientes (detectar que existem, avisar que o servidor encerrou, etc.) a funcionar.
- Lançamento e comunicação com o dicionário preparada, incluindo, para testes, a verificação de uma palavra que existe e uma que não existe.
- *Login* dos utilizadores a funcionar.

Para esta meta é admissível assumir que só existe um cliente em simultâneo.

Data de entrega: Domingo, 2 de Dezembro. Sem possibilidade de entrega atrasada.

Meta 3:

Requisitos:

- Todos os requisitos expostos no enunciado.

Data de entrega: Domingo, 1 de Janeiro, 2019. Sujeito a ajustes caso haja alterações no calendário escolar. Entrega atrasada possível, mas condicionada e fortemente penalizada.

Em todas as metas deverá ser entregue um relatório. O relatório compreenderá o conteúdo que for relevante para justificar o trabalho feito, deverá ser da total autoria dos alunos, e deverá ainda compreender um conteúdo obrigatório que será especificado no *moodle* atempadamente.

6. Avaliação do trabalho

Para a avaliação do trabalho serão tomados em conta os seguintes elementos:

- **Arquitectura do sistema** – Há aspectos relativos à interação dos vários processos que devem ser cuidadosamente planeados de forma a se ter uma solução elegante, leve e simples. A arquitectura deve ser bem explicada no relatório para não existirem mal-entendidos.
- **Implementação** – Deve ser racional e não deve desperdiçar recursos do sistema. As soluções encontradas para cada problema do trabalho devem ser claras. O estilo de programação deve seguir as boas práticas. O código deve ter comentários relevantes. Os recursos do sistema devem ser usados de acordo com a sua natureza.
- **Relatório** – Em todas as metas existirá um relatório. O relatório seguirá indicações a fornecer no *moodle* atempadamente. De forma geral, o relatório descreve a estratégia e os modelos seguidos, a estrutura da implementação e as opções tomadas. Este relatório será enviado juntamente com o código no arquivo submetido via *moodle* e o da entrega final será também entregue em mão (i.e., impresso) na defesa.
- **Defesa** – Os trabalhos são sujeitos a defesa individual onde será testada a autenticidade dos seus autores. Pode haver mais do que uma defesa caso subsistam dúvidas quanto à autoria dos trabalhos. A nota final do trabalho é diretamente proporcional à qualidade da defesa. Elementos do mesmo grupo podem ter notas diferentes consoante o empenho individual que demonstraram na realização do trabalho.

  Apesar da defesa ser individual, ambos os elementos do grupo devem comparecer ao mesmo tempo. A falta à defesa implica automaticamente a perda da totalidade da nota do trabalho.

  Na FUC está descrito o que acontece nas situações de fraude (ex., plágio) no trabalho.
- Os trabalhos que não funcionam são fortemente penalizados independentemente da qualidade do código-fonte apresentado. Trabalhos que nem sequer compilam terão uma nota extremamente baixa.
- A identificação dos elementos de grupo deve ser clara e inequívoca (tanto no arquivo como no relatório). Trabalhos anónimos não são corrigidos.
- Qualquer desvio quanto ao formato e forma nas submissões (exemplo, tipo de ficheiro) dará lugar a penalizações.