Sistemas Operativos 19/20

Trabalho Prático - Programação em C para UNIX

O trabalho prático de sistemas operativos consiste na implementação de um sistema de gestão e redistribuição de mensagens denominado **MSGDIST**. As mensagens são simples mensagens de texto, descritas com mais detalhe adiante. O sistema encarrega-se de aceitar mensagens, armazená-las e redistribuí-las a quem estiver interessado nelas. Os detalhes são dados mais adiante.

O trabalho prático deve ser realizado em linguagem C, para Unix (Linux), usando os mecanismos do sistema abordados nas aulas teóricas e práticas. Deve ser dada prioridade ao uso de chamadas ao sistema operativo (documentadas na secção 2 das *manpages*) face ao uso de funções biblioteca (documentadas na secção 3 das *manpages*). O trabalho foca-se no uso correto dos mecanismos e recursos do sistema e não é dada particular preferência a detalhes de escolhas na implementação de aspetos de caráter periférico à disciplina (por exemplo: a pergunta "devo usar uma lista ligada ou um *array* dinâmico?" terá como resposta um encolher de ombros). O recurso a bibliotecas que façam ou ocultem parte do trabalho não é permitido. O recurso a mecanismos do sistema que não tenham sido abordados nas aulas é permitido, mas terão que ser devidamente explicados. Não é permitida uma abordagem baseada na mera colagem de excertos de outros programas ou exemplos. Todo o código apresentado terá que ser entendido e explicado por quem o apresenta.

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

1. Descrição geral

O sistema MSGDIST compreende os seguintes elementos:

- Pessoas: Administrador e Utilizadores
- Processos e Programas: Gestor, Verificador, Clientes

Estão envolvidos os seguintes conceitos:

- Mensagem e tópico.
    - Uma mensagem é um texto organizado em campos: tópico, título, corpo, duração. O corpo da mensagem pode ter até 1000 caracteres. Defina o tamanho máximo para a mensagem no total (com todos os seus campos) conforme entender.
    - Um tópico é um tema abrangente (assunto) no qual as mensagens podem ser enquadradas. Uma mensagem enviada por um cliente pode mencionar um tópico novo, i.e., inexistente no gestor. Neste caso, o gestor passará a contar com mais um tópico na sua lista de tópicos disponíveis para subscrição.
    - A duração é o tempo máximo que a mensagem ficará armazenada no sistema (no processo gestor).
    - Toda a informação aqui referida é textual exceto a duração da mensagem que é um inteiro que representa o número de segundos de duração da mensagem.

O objetivo do sistema MSGDIST é mediar a receção e entrega de mensagens entre clientes (em benefício dos seus respetivos utilizadores). Os clientes enviam mensagens ao gestor que serão depois entregues aos clientes consoante as escolhas e consultas que estes fazem. A funcionalidade é bastante direta e não são necessários mecanismos mais elaborados do que o descrito.

Formas de uso do cliente

Todos os utilizadores usam o mesmo programa cliente para interagir com o gestor. Cada utilizador executará uma instância (i.e., processo) do programa cliente. Através do programa cliente, o utilizador pode:

- Escrever uma nova mensagem, especificando todos os seus campos, e enviá-la ao gestor. A elaboração da mensagem deverá seguir um formado de edição de um campo de texto (N linhas x M colunas, com N e M escolhidos de forma a permitir os 1000 caracteres) em que o utilizador pode movimentar o cursor e escrever onde quiser, de uma forma semelhante a um "notepad". Isto pode ser feito usando a biblioteca *ncurses*.
    - A mensagem, depois de enviada e aceite pelo gestor, fica disponível aos demais utilizadores. Se um utilizador estiver a usar um programa cliente, e tiver subscrito o tópico da mensagem enviada, deve ser imediatamente alertado pelo seu cliente em resposta à notificação que este último recebe do gestor.
- Consultar a lista de tópicos atualmente existente (tenham ou não mensagens).
- Consultar a lista de títulos de mensagens de um determinado tópico.
- Consultar uma mensagem de um tópico.
- Subscrever / cancelar subscrição de um tópico.

O cliente é alertado sempre que uma nova mensagem de um tópico por ele subscrito fica disponível no gestor. Nesta situação é apresentada no ecrã a mensagem "nova mensagem (título da mensagem) do tópico (nome do tópico) disponível durante (duração)". Este aviso aparece no ecrã no instante que o gestor envia o aviso, independentemente do que o cliente esteja a fazer nesse momento, e de uma forma que não interfira com o que o utilizador desse cliente esteja a fazer nesse momento (por exemplo a escrever outra mensagem). Assim, deverá ser tido algum cuidado na elaboração da interface com o utilizador (sugestão: recorrer à biblioteca *ncurses*).

O utilizador indica o seu *username* ao programa cliente como argumento da linha de comandos deste último. O primeiro passo do programa cliente é identificar o seu utilizador perante o gestor. Isto corresponde a enviar o *username* indicado. Se nesse instante estiver outro utilizador com esse *username* a participar no sistema MSGDIST, então o gestor acrescenta automaticamente um número ao *username* (ex., "manuel" passa a "manuel3" porque já existem neste momento o "manuel", "manuel1" e "manuel2"). O cliente é informado sempre acerca do *username* que efetivamente lhe coube. Quando houver diferença entre o *username* mencionado pelo utilizador na sua linha de comandos e aquele que o servidor lhe atribuiu, o cliente deve informar o seu utilizador.

Interação automática entre o cliente e o servidor:

- O cliente recusar-se-á a prosseguir a sua execução se perceber que o gestor não está em execução (Nota: há diversas formas de concretizar este aspeto).
- Quando o utilizador encerra o seu processo cliente, este último deverá informar o gestor deste facto, para que se liberte o seu *username* e cesse o envio de avisos de novas mensagens em tópicos subscritos por esse utilizador.
- Ao terminar, o gestor informará os clientes conhecidos até esse momento, e que se encontram em execução, que vai terminar. Os clientes deverão também terminar, informando o utilizador desse fato.
- O gestor detetará automaticamente que o cliente deixou de existir com uma precisão de 10 segundos (ou seja, se o cliente terminar abruptamente (por exemplo, foi terminado pelo sistema por uso indevido de memória), o gestor saberá isso o mais tardar 10 segundos depois e tratará esta situação como um encerramento de cliente).
- O cliente detetará que o gestor deixou de estar a correr com uma precisão de, no máximo, 10 segundos. Este requisito e o anterior são fáceis de realizar recorrendo a mecanismos de *heartbeat* / *keep alive* (investigar por estas palavras chave o conceito subjacente e desenvolver com base nos mecanismos abordados na unidade curricular – existem diversas formas de concretizar este mecanismo). Este ponto está relacionado com o anterior: os mecanismos usados pelo gestor e pelo cliente são planeados de forma a funcionar em conjunto.

A troca de informação entre cliente e servidor é feita por *named pipes*. Podem existir outros mecanismos envolvidos na interação entre cliente e servidor, mas relativamente à informação trocada, deverão ser usados *named pipes*. Não existe qualquer interação direta entre dois clientes.

Funcionalidade do gestor

O gestor deverá concretizar a funcionalidade necessária para cumprir aquilo que os clientes esperam dele, tal como descrito acima:

- Recebe mensagens enviadas pelos clientes, armazena-as durante o tempo indicado no campo duração da mensagem e elimina-as automaticamente quanto a duração se esgota. O número máximo de mensagens a armazenar é fixo durante a execução do gestor e é fornecido pelo valor da variável de ambiente MAXMSG, consultada no arranque do gestor.
- Avisa os clientes/utilizadores acerca da existência de novas mensagens recebidas nos tópicos por eles subscritos.
- Mantém informação acerca de que clientes e utilizadores se encontram neste momento a interagir com ele. Apesar de relacionados na proporção de 1 cliente – 1 utilizador, cliente e utilizador são entidades distintas. Cliente é um processo e utilizador a pessoa que instanciou o programa cliente e que é identificada no sistema por um *username* específico em determinado instante.
- Mantém a informação acerca de que tópicos estão subscritos por que clientes/utilizadores.
- É lançado pelo administrador e aceita, por parte deste, comandos escritos, introduzidos segundo o paradigma da "linha de comandos" (ou seja, não devem ser usados menus).
- Deteta que clientes terminaram a sua execução (porque os respetivos utilizadores os encerraram, ou porque deixaram de dar sinais de vida por mais de 10 segundos).
- Avisa os clientes conhecidos que vai terminar, e permite que o cliente saiba que ainda está em execução (conforme essa parte da funcionalidade for implementada no lado do cliente).
- Reporta de imediato no *stderr* todas as ações feitas que alterem a lista de mensagens ou tópicos (alguns exemplos: mensagem recebida, eliminação das mensagens publicadas quando as mesmas expirarem, a criação de novos tópicos e a eliminação de tópicos, etc.), e as ações que alterem o conjunto de clientes/utilizadores (alguns exemplos: utilizador entrou, cliente deixou de existir, etc.). O reporte consiste em mensagens de texto com indicação do que aconteceu. Estas mensagens são enviadas para o *stderr*.

Cada mensagem recebida pelo gestor é analisada com o auxílio do programa independente "verificador", que recebe o corpo da mensagem e identifica o número de palavras proibidas nele existente. Se o número de palavras proibidas exceder o valor indicado na variável de ambiente MAXNOT, então a mensagem é rejeitada e o cliente que a indicou é avisado desse facto. As palavras proibidas estão no ficheiro de texto cujo nome está na variável de ambiente WORDSNOT. Repare, ao analisar o código (fornecido) do verificador, a forma como este programa obtém o nome desse ficheiro.

O código do programa verificador é fornecido em anexo. Deverá compilar esse programa de forma que o executável fique disponível ao gestor. Deve analisar o código fonte fornecido para perceber como é que o "verificador" obtém as palavras a verificar e de que forma indica a contabilização resultante de modo a perceber como deve o gestor comunicar com ele. Também deve experimentar executá-lo em linha de comandos. Não pode alterar rigorosamente nada no código fonte do programa "verificador". O gestor deverá procurar otimizar a utilização do verificador (por exemplo, mantê-lo em execução em vez de estar sempre a executar uma nova instância a cada nova mensagem recebida).

Persistência de dados

Não existe qualquer persistência de informação de uma execução do gestor para outra nem da execução de um cliente para outra pelo mesmo utilizador:

- Não existe registo permanente em ficheiro acerca de tópicos subscritos por que cliente ou utilizador. Assim que o cliente encerra o gestor esquece a sua existência.
- Não existe conta permanente de utilizador (nem sequer há *passwords*).
- As mensagens no gestor existem apenas em memória. Quando o gestor encerra, todas as mensagens, tópicos e informação sobre utilizador é perdida.
- O utilizador terá que subscrever novamente os tópicos que lhe interessam a cada nova execução do cliente.

Se quiser, pode tornar esta informação persistente (em ficheiro). Mas não faz parte dos requisitos.

Comportamento automático do gestor

O gestor deve efetuar toda a verificação da duração temporal das mensagens, a notificação aos clientes de novas mensagens, a verificação da existência dos clientes e o que for necessário para que os clientes percebam que o gestor ainda existe. Esta lista aqui apresentada não é exaustiva e deve deduzir o restante comportamento automático com base no resto do texto.

Administração do gestor

O administrador é um utilizador que interage com o gestor através de comandos escritos. A lógica é semelhante à da linha de comandos de uma *shell*. No entanto, trata-se de uma analogia: não deve usar a *bash* - simplesmente não deve usar menus. A intenção do administrador é expressa através de uma linha de texto com várias palavras em que a primeira é o comando e as seguintes são as opções ou valores desse comando. A linha é lida de uma só vez: uma linha – uma ação que o administrador deseja ver cumprida.

As ações que o administrador pode desencadear são:

- Ligar/desligar a filtragem de palavras proibidas: **filter on** / **filtro off**
- Listar utilizadores: **users**
- Listar tópicos: **topics**
- Listar mensagens: **msg**

- Listar mensagens de um tópico: **topic *tópico-em-questão***
- Apagar mensagem (tem que propor e adotar uma forma de as identificar): **del *mensagem-em-questão***
- Excluir um utilizador (que será avisado e cujo cliente encerrará): **kick *username-em-questão***
- Desligar o gestor: **shutdown**
- Eliminar tópicos que não tenham nenhuma mensagem de momento (as subscrições desses tópicos são canceladas e os respetivos utilizadores são avisados): **prune**

Pode implementar outras opções à escolha (sugestões: listar mensagens de um determinado utilizador, listar mensagens a expirar nos próximos x segundos, etc.). Não são extras, mas talvez compensem alguma outra falha e podem dar jeito para *debug*.

Os seguintes mecanismos devem poder funcionar de forma independente / paralela uns dos outros de forma a que o atraso num deles não ponha diretamente em causa a execução dos restantes.

- Verificação de palavras através do verificador
- Processamento de mensagens enviadas pelos clientes
- Atendimento dos comandos do administrador
- Deteção de clientes em encerramento ou incontatáveis
- Verificação das idades das mensagens e eliminação das que expiraram.

Alguns reparos

O gestor tem que conseguir lidar com várias tarefas em simultâneo. O cliente também. A lista de tarefas simultâneas no gestor não é a mesma que a do cliente.

Interface visual e perceção visual da edição

A interface visual é mais importante no cliente para o caso da edição do texto da mensagem e para a apresentação dos avisos do gestor. Pode usar a biblioteca *ncurses*. A biblioteca *ncurses* contém diversas funções para impressão de caracteres, leitura de informação via teclado, posicionamento de 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*". Não há muito mais a dizer sobe isso. Eventualmente serão dadas algumas indicações nas aulas. Entre outros, pode consultar o site

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

2. Considerações adicionais

- Os programas cliente e gestor são lançados através da linha de comandos. O verificador não é lançado diretamente pelo utilizador. 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.

- Apenas deve existir um processo gestor em execução. Caso seja executada uma nova instância do gestor, esta deverá detetar a existência da primeira e terminar de imediato.
- Caso existam diversas tarefas feitas em simultâneo envolvendo os mesmos dados, deverá haver o cuidado de sincronizar o acesso aos mesmos usando mecanismos adequados de modo a assegurar sempre a consistência da informação manipulada.
- Podem existir diversas situações e potenciais situações de conflito ou erro 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 de lidar com o cenário.

  Caso se detetem incoerências neste enunciado, as mesmas serão corrigidas e publicadas. Poderá haver lugar a documentos adicionais no *moodle*, nomeadamente um *Frequently Asked Questions* (FAQ) com as perguntas mais frequentes e respetivas respostas.

3. 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 dois alunos (grupos de três não são admitidos e qualquer pedido nesse sentido será sempre negado ou ignorado).
- Existe defesa obrigatória. A defesa será efetuada em moldes a definir e anunciados através dos canais habituais na altura em que tal for relevante.
- Existem duas 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_1920_tp_meta1_*nome1_numero1_nome2_numero2*.zip

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

  A não aderência ao formato de compressão indicado ou ao padrão do nome do ficheiro será penalizada, podendo levar a que o trabalho nem seja visto.

[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**.

Metas: requisitos e datas de entrega

Meta 1:

Requisitos:

- Planear e definir as estruturas de dados responsáveis por gerir as definições de funcionamento no gestor e no cliente. Definir os vários *header files* com constantes simbólicas que registem os valores por omissão comuns e específicos do cliente e servidor bem como as estruturas de dados relevantes.
- Desenvolver a lógica de leitura das variáveis de ambiente do gestor e do cliente, refletindo-se nas estruturas de dados mencionadas no ponto anterior. Sugestão: usar as funções `getopt()`, `getsubopt()` e `getenv()`.
- Iniciar o desenvolvimento da leitura de comandos de administração do gestor implementando a leitura e validação dos comandos e respetivos parâmetros e a implementação completa do comando `shutdown`.
- Preparar a ligação entre gestor e verificador de forma a permitir testar a funcionalidade do filtro com algumas palavras enviadas a partir do gestor.
- Desenvolver e entregar um *makefile* que possua os *targets* de compilação "all" (compilação de todos os programas), "cliente" (compilação do programa cliente), "gestor" (compilação do programa servidor), "verificador" (compilação do programa verificador) e "clean" (eliminação de todos os ficheiros temporários de apoio à compilação e dos executáveis).

Data de entrega: Domingo, 17 de Novembro. Sem possibilidade de entrega atrasada.

Meta 2:

Requisitos:

- Todos os requisitos expostos no enunciado.

Data de entrega: Domingo, 5 de Janeiro, 2020. Sujeito a ajustes caso haja alterações no calendário escolar.

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 exclusiva autoria dos membros do grupo, e deverá ainda compreender um conteúdo obrigatório que será especificado no *moodle* atempadamente.

4. Avaliação do trabalho

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

- **Arquitetura do sistema** – Há aspetos relativos à interação dos vários processos que devem ser cuidadosamente planeados de forma a apresentar-se uma solução elegante, leve e simples. A arquitetura 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 e bem documentadas no relatório. 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** – Os relatórios seguirão as indicações a fornecer no *moodle*. De forma geral, os relatórios descrevem a estratégia e os modelos seguidos, a estrutura da implementação e as opções tomadas. Serão enviados juntamente com o código no arquivo submetido via *moodle* da meta em questão.
- **Defesa** – Os trabalhos são sujeitos a defesa individual onde será testada a autenticidade da sua autoria. 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 desempenho individual que demonstraram na defesa.

  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.

  Este ano será usado um software que automatiza a deteção de fraude dos trabalhos entregues. Na FUC está descrito o que acontece nas situações de fraude (ex., plágio) no trabalho.
- Os trabalhos que não funcionem serão fortemente penalizados independentemente da qualidade do código-fonte ou arquitetura apresentados. 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.