Instituto Superior de Engenharia de Coimbra
Departamento de Engenharia Informática e de Sistemas

Sistemas Operativos 20/21

Trabalho Prático - Programação em C para UNIX

O trabalho prático de sistemas operativos consiste na implementação de um sistema de gestão de campeonatos de jogos denominado **CHAMPION**. Os jogos envolvidos são muito simples e não são o alvo direto do trabalho. O sistema pretendido encarrega-se de fazer a ponte entre os jogadores e os jogos, mediando as mensagens trocadas entre ambos e gerindo o campeonato. Os detalhes precisos são dados mais adiante.

O trabalho prático deve ser concretizado em linguagem C, para plataforma Unix (Linux), usando os mecanismos do sistema operativo abordados nas aulas teóricas e práticas. No que respeita à manipulação de recursos do sistema deve ser dada prioridade ao uso de chamadas ao sistema operativo[1] face ao uso de funções biblioteca[2] (por exemplo, devem ser usadas *read()* e *write()* em vez de *fread()* e *fwrite()*). 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 de 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 gestão de campeonatos de jogos.

[1] Documentadas na secção 2 das *manpages*
[2] Documentadas na secção 3 das *manpages*

1. Descrição geral

Existem vários jogos e vários jogadores. Todos os jogos são *single-player* (não existe a situação de um jogador contra outro) e todos exibem uma lógica de interação semelhante. Será lançado um campeonato que decorre durante alguns minutos. Durante o campeonato, cada jogador pode interagir com um jogo apenas, sendo-lhe atribuída uma pontuação quando o campeonato termina. Jogadores diferentes podem eventualmente estar a jogar jogos diferentes. A atribuição jogador-jogo não é controlada pelo jogador, mas este saberá qual o jogo que lhe foi atribuído. Aspetos como dificuldade do jogo são perfeitamente irrelevantes no trabalho, e se um jogo for mais difícil do que outro, tanto pior para o jogador a quem foi atribuído.

O objetivo do sistema CHAMPION é mediar a comunicação entre as aplicações cliente usadas pelos jogadores e os jogos propriamente ditos, gerindo o campeonato e mantendo uma tabela de pontuações.

2. Conceitos e entidades

O trabalho envolve os seguintes conceitos:

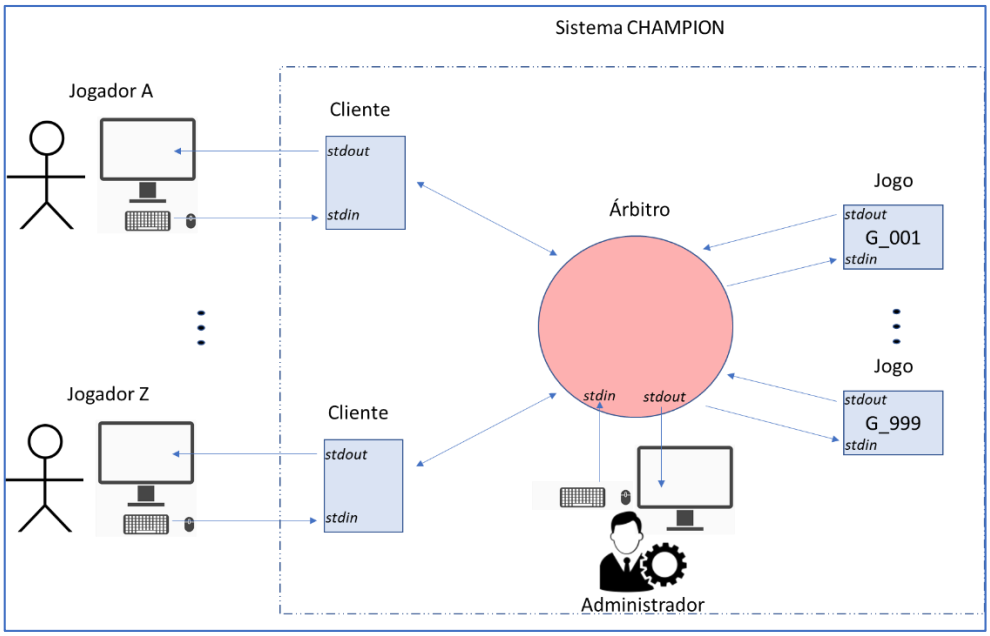
- Jogo
    - O "jogo" é um programa de caracter secundário que implementa um jogo qualquer cuja lógica específica não é importante. Exemplos de jogos possíveis:
        - Adivinhar o número que o computador "pensou" (i.e., gerou aleatoriamente). Se o jogador acertar, o jogo sorteia um novo número e o jogo continua.
        - Identificar qual a tradução em português de uma palavra noutra língua (palavra sorteada de uma qualquer lista *built-in* ao programa);
        - Identificar o resultado de uma expressão aritmética "complicada";
        - Etc.
    - O jogo decorre com uma sequência de perguntas-respostas entre o programa e o jogador. Assim, usando o exemplo da tradução de palavras mencionado acima, seria apresentado o desafio ("O que significa "XXXX"?"), seria recebida a resposta, a qual estaria certa ou errada, e em seguida o jogo passa à próxima pergunta ("O que significa "YYYY"?").
    - A duração dos jogos é igual à duração do campeonato: não existe propriamente um objetivo que ao ser atingido pelo jogador o faria terminar. No exemplo dado acima de adivinhar um número, se o jogador acertar e o campeonato ainda não tiver terminado, o jogo sorteia outro número e o jogo continua.
    - Haverá uma pontuação, completamente decidida pelo jogo em questão. No exemplo da tradução de palavras, poderia ser o número de traduções certas menos o número de traduções erradas conseguidas durante o período de jogo.

- Aspetos de carácter obrigatório:
  - O *output* do jogo (por exemplo, a "pergunta desafio") é efetuado pelo seu *stdout* e o *input* (a resposta do jogador) é recebido no seu *stdin*.
  - A pontuação obtida é comunicada pelo jogo no seu final através do seu *exit status*.
- Existirão vários jogos. Os ficheiros executáveis que lhes correspondem têm um nome que começa sempre por "g_". Estes encontram-se na diretoria indicada pela variável de ambiente GAMEDIR. Se esta não estiver definida deve ser assumida a diretoria atual.

- Campeonato
  - Um "campeonato" representa um conjunto de jogos que são jogados durante um determinado intervalo de tempo. O campeonato é aberto a um número máximo de jogadores em simultâneo, especificado pela variável de ambiente MAXPLAYERS e que se sabe à partida que nunca será superior a 30. Os jogos são todos *single-player*. O jogador joga um e um só jogo durante o campeonato (o jogo envolverá várias interações com o jogador). O jogador não escolhe o jogo – é-lhe atribuído pelo árbitro do sistema.

- Cliente
  - Um "cliente" é um programa usado pelo jogador para interagir com o árbitro do sistema CHAMPION e, indiretamente, com o jogo. O cliente não implementa jogo nenhum: apenas recebe o *input* do jogador, encaminhando-o para o árbitro, e recebe a informação oriunda deste último (e, indiretamente, do jogo), apresentando-a no ecrã. O *input* recebido do cliente pelo árbitro tanto pode ser informação destinada ao jogo que o jogador esteja a jogar como ao próprio árbitro, sendo a distinção da responsabilidade deste último.

- Árbitro do sistema CHAMPION
  - O "Árbitro" do sistema CHAMPION é a entidade que está entre o cliente e os jogos. Este programa árbitro é gerido por um administrador. O jogador não vê diretamente o árbitro, interagindo apenas com o seu cliente, o qual interage com o árbitro, o qual por sua vez interage com os jogos.



O sistema CHAMPION compreende os seguintes elementos:

- Jogadores (vários)
  - São identificados por um nome e não poderá haver repetições. Não existe password.
  - Os jogadores existem apenas no contexto do sistema CHAMPION, não havendo qualquer relação com os utilizadores do sistema operativo.
  - No mesmo campeonato não pode haver dois jogadores com o mesmo nome.

- Administrador
  - Efetua algumas ações de gestão do campeonato. Não joga. Não tem *username* associado nem *password*. É simplesmente o utilizador que interage com o processo que gere o campeonato (o árbitro).

O sistema CHAMPION engloba os seguintes programas:

- Cliente (já descrito acima)
  - Cumpre a interface com o jogador, que segue o paradigma de consola (texto). O uso de menus não é aceite. Não será necessário recorrer a qualquer composição gráfica.
  - Cada jogador executa um cliente.

- Árbitro
  - Gere o campeonato.
  - Existe apenas um processo Árbitro a correr em simultâneo e este aspeto deve ser garantido pelo seu código.
  - As suas funções incluem

- Criar os mecanismos de comunicação para:
  - Receber dos clientes pedidos de participação e mensagens destinadas aos jogos.
  - Receber/enviar informação aos jogos.
- Manter em memória e gerir os dados dos utilizadores atualmente em campeonato, incluindo as pontuações.
- Interagir com o administrador e cumprir as suas instruções. Esta interação é feita segundo o paradigma de consola (texto) por comandos escritos (nada de "menus").
- Interagir com os jogos, gerindo o seu funcionamento onde tal seja necessário.
- Interagir com os clientes.
- Gerir o campeonato, garantindo as suas características operacionais (duração, jogadores admitidos, etc.).

- Jogos
  - São os programas descritos no início deste enunciado. É obrigatório implementar pelo menos um, diferente dos exemplos citados acima, e cujas regras cada grupo tem inteira liberdade de definir. Recomenda-se que qualquer jogo no início apresente uma mensagem de boas vindas ao utilizador e um breve sumário suficiente para o jogador conhecer o nome do jogo e perceber as regras do mesmo. Repare que pode testar o jogo sem que o sistema CHAMPION esteja desenvolvido uma vez que ele usa a sua *stdin*/*stdout* para interagir com o exterior (i.e., o jogador).

3. Características operacionais de um campeonato

O campeonato fica pronto assim que o Árbitro é lançado. O programa recebe por argumento da linha de comando alguns dados de configuração e analisa as variáveis de ambiente GAMEDIR e MAXPLAYERS. Caso algo não esteja de acordo com os pressupostos o programa deverá sair de forma ordeira.

O Árbitro aguarda que pelo menos dois jogadores estejam prontos. Após dois jogadores já estarem presentes, aguardará um tempo máximo por mais jogadores, findo o qual começa o campeonato.

O início do campeonato envolve atribuir um jogo (escolhido pelo árbitro de forma aleatória de entre os existentes) a cada jogador. O jogador é informado de qual o jogo que lhe foi atribuído. O árbitro é responsável por preparar a execução dos vários jogos e encaminhar (intermediar) toda a informação entre cliente do jogador e jogo que lhe foi atribuído nesse campeonato. O árbitro mantém toda a informação relativa aos jogadores, atualizando sempre que há alterações (novo jogador, alguém saiu, recebida pontuação, etc.)

O campeonato tem uma duração máxima. Findo este tempo: os jogos são avisados que devem terminar através de um sinal SIGUSR1; os jogos comunicam ao árbitro a pontuação do "seu" jogador através do *exit status*, tal como já descrito; o sinal SIGUSR1 é também enviado aos clientes para os informar que o campeonato terminou (isto não significa que os clientes devam terminar); o árbitro informará todos os clientes acerca da pontuação do seu jogador e também divulgará a identidade do jogador vencedor (i.e., que totalizou mais pontos naquele campeonato).

Findo um campeonato, o árbitro mantém-se em execução e o ciclo repete-se. Os clientes interessados em jogar de novo não precisam de terminar e voltar a executar, mas terão que repetir o processo de interação com o árbitro a partir do início (identificar jogador, etc.).

Durante o campeonato presume-se que a informação que é enviada do cliente ao árbitro é para reencaminhar ao jogo que lhe está atribuído. Esta informação é texto, de acordo com a lógica de interação com o jogo tal como descrito atrás. Qualquer informação enviada pelo cliente que seja para consumo interno do árbitro deverá ser precedida por "#". Encaixam-se nesta situação o pedido para obter a identificação do jogo na forma "#mygame" (usado pelo utilizador caso já se tenha esquecido, pois essa informação é-lhe dada no início do campeonato), e o pedido para indicar que desiste do campeonato – "#quit". Na sequência do segundo pedido o jogo atribuído a este jogador deverá também terminar. Se o campeonato ficar apenas com um jogador ativo deve terminar imediatamente, cabendo a vitória ao último jogador.

Os jogos são atribuídos aos jogadores de forma aleatória e pode haver dois ou mais jogadores a executar o mesmo jogo (mas em processos diferentes). São considerados jogos todos os programas que existem na diretoria indicada pela variável de ambiente GAMEDIR e que começam por "g_". O administrador poderá interferir com o processo de atribuição de jogos a jogadores.

A duração do campeonato e o tempo de espera por mais jogadores para além dos dois primeiros são indicados ao árbitro por argumentos de linha de comandos aquando do seu lançamento.

Sempre que o árbitro sair, todos os clientes deverão ser informados. Os recursos que já não são necessários não devem permanecer em memória.

4. Formas de uso

Uso do cliente

O cliente deverá

- Obter a identificação do jogador.
- Recolher as ordens do jogador, relativas ao jogo ou outras de controlo (desistir, outras).
- Apresentar os dados relativos ao decorrer do jogo e a informação relativa ao início e final de um campeonato, assim como outras informações de controlo.
- A interação com o utilizador deve ser realizada em "modo texto" por comandos escritos. Aquilo que o jogador deverá escrever é dependente da lógica do jogo que lhe calhar. Não é necessário gerir a formatação de conteúdo (posicionamento, cores, etc.).

Uso do árbitro

O árbitro interage apenas com o administrador segundo a lógica de comandos escritos. Este terá à sua disposição comandos para:

- Listar jogadores em jogo (nome e jogo atribuído). Comando "players"
- Listar jogos disponíveis. Comando "games".
- Remover um jogador do campeonato. O efeito é o mesmo que o jogador desistir, mas neste caso o cliente é informado do que aconteceu. Comando "k" concatenado com o nome do jogador (ex. "krui" – remove jogador "rui").
- Suspender a comunicação entre jogador e jogo. As mensagens deste jogador-jogo deixam de ser encaminhadas entre ambos. O cliente em questão é informado do que se passa. Comando "s" concatenado com o nome do jogador.
- Retomar a comunicação entre jogador e jogo. As mensagens deste jogador-jogo voltam a ser encaminhadas. O cliente em questão é informado. Comando "r" concatenado com o nome do jogador.
- Encerrar o campeonato imediatamente. Tem o mesmo efeito que ocorre quando o campeonato chega ao fim do seu tempo. Comando "end".
- Sair, encerrando o árbitro. Comando "exit".

Podem, opcionalmente, ser implementados outros comandos (sem valorização atribuída), para efeitos de teste e depuração. Por exemplo, um comando que force a atribuição de um jogo específico a um jogador.

5. Restrições de implementação

Não existe comunicação direta entre o cliente e os jogos.

Só podem ser usados os mecanismos de comunicação que foram abordados nas aulas. *Pipes* e *named pipes* terão um papel preponderante. Em momento algum poderá usar ficheiros regulares como mecanismo de comunicação entre processos.

Deve precaver a validade das operações efetuadas pelos programas, de acordo com a estratégia que for implementada. Esta questão decorre das características da solução que é escolhida e não de algo que seja dito no enunciado. Por exemplo, se em algum momento a estratégia seguida necessitar de mecanismos de sincronização, então essa necessidade deve ser garantida, não podendo ser invocada a lógica "o enunciado não fala sobre isso" ou "isso é uma situação que raramente acontecerá".

Persistência de dados

Não existe qualquer persistência de informação entre execuções distintas do árbitro, e o mesmo se aplica às execuções dos clientes. Não existe registo permanente de utilizadores nem de pontuações.

Se quiser, pode tornar esta informação persistente (em ficheiro), mas tal é opcional e sem valorização associada.

6. Considerações adicionais

- Toda a ação do sistema aqui descrito decorre na mesma máquina e na mesma conta de utilizador. Serão usadas consolas/terminais diferentes para cada cliente, mais uma para o árbitro.
- O cliente tem que conseguir lidar com várias tarefas em simultâneo. O árbitro também. Deve pensar nisto.
- 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.

7. 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 deve 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. A defesa será presencial exceto se houver ordens em contrário pela presidência da escola.
- 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***[3] cujo **nome** respeita o seguinte padrão[4]:

  so_2021_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 adesão ao formato de compressão indicado ou ao padrão do nome do ficheiro será penalizada, podendo levar a que o trabalho nem seja visto.

Metas: requisitos e datas de entrega

Meta 1: **8 de Novembro**

Requisitos:

- Planear e definir as estruturas de dados responsáveis por gerir as definições de funcionamento no árbitro 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.
- Obter a "duração do campeonato" e o "tempo de espera", aquando da execução do árbitro, através da leitura dos argumentos de linha de comandos. Sugestão: use as funções `getopt()` e `getsubopt()`.
- Implementar um jogo respeitando as características referidas no enunciado, exceto o que se refere ao tratamento de sinais, que ainda não terá que existir. Bastará algo muito simples desde que siga a lógica dos jogos a usar com o sistema CHAMPION.
- Desenvolver a lógica de leitura das variáveis de ambiente GAMEDIR e MAXPLAYER pelo árbitro, refletindo-se nas estruturas de dados mencionadas no ponto anterior. Sugestão: use a função `getenv()`.

[3] Leia-se "***zip***" - não é *arj*, *rar*, *tar*, ou outros. O uso de outro formato poderá ser **penalizado**. Há muitos utilitários da linha de comando UNIX para lidar com estes ficheiros (zip, gzip, etc.). Use um.
[4] O não cumprimento do formato do nome causa atrasos na gestão dos trabalhos recebidos e será **penalizado**.

- Desenvolver e entregar um *makefile* que possua os *targets* de compilação "all" (compilação de todos os programas), "cliente" (compilação do programa cliente), "árbitro" (compilação do programa servidor), "jogo" (compilação do programa jogo proposto) e "clean" (eliminação de todos os ficheiros temporários de apoio à compilação e dos executáveis).

Data de entrega: Domingo, 8 de Novembro, 2020. Sem possibilidade de entrega atrasada.

Meta 2: 13 de Dezembro

Requisitos:

- Acrescentar ao jogo já implementado na meta 1 a parte do término aquando da receção do sinal SIGUSR1 e a disponibilização do resultado através do *exit status*.
- Implementar toda a lógica de comunicação entre os clientes e o árbitro através do mecanismo de *named pipes*.
- Implementar os comandos suportados pelo cliente: `#mygame` e `#quit`.
- Iniciar o desenvolvimento da leitura de comandos de administração do árbitro implementando a leitura e validação dos comandos e respetivos parâmetros, incluindo a implementação completa dos comandos `players`, `games`, `k` e `exit`.

Data de entrega: Domingo, 13 de Dezembro, 2020. Sem possibilidade de entrega atrasada.

Meta 3: 24 de Janeiro

Requisitos:

- Todos os requisitos expostos no enunciado.

Data de entrega: Domingo, 24 de Janeiro, 2021. Sujeito a ajustes caso haja alterações no calendário escolar.

Nas **metas 1** e **2** deverá ser entregue um documento com duas páginas descrevendo os pormenores da implementação e principais opções tomadas.

Na **meta 3** 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. Caso venha a ser divulgado entretanto um guia de elaboração do relatório, então este deverá seguir as indicações dadas.

8. 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** – O relatório deve descrever convenientemente o trabalho. Descrições meramente superficiais ou genéricas de pouco servirão. De forma geral, o relatório descreve a estratégia e os modelos seguidos, a estrutura da implementação e as opções tomadas. Podem vir a ser dadas indicações adicionais sobre a sua elaboração. O relatório deve ser entregue juntamente com o código no arquivo submetido via *moodle* da meta em questão.
- **Defesa** – Os trabalhos são sujeitos a defesa individual, durante a qual será verificada a respetiva autenticidade. 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 e grau de participação individuais 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.