Sistemas Operativos 16/17

Trabalho Prático - Programação em C para Unix

O trabalho prático consiste na implementação do jogo de futebol multijogador. A interface será em consola/modo texto, e todos os jogadores estarão ligados à mesma máquina UNIX, ou seja, não se pretende a utilização de mecanismos de comunicação em rede – apenas mecanismos de comunicação inter-processo.

1. Descrição geral

O trabalho prático consiste na implementação de um simulador de jogo de futebol simplificado. Acerca do jogo de futebol é de salientar já de início o seguinte:

- Trata-se de uma simplificação dos jogos reais.
- Não se pretendem implementar as regras de futebol da realidade (foras de jogo, penalty, etc.).
- Não haverá necessariamente tantos jogadores em campo como nos jogos reais.

➔ Para facilitar a leitura, definem-se aqui dois termos que serão usados no resto do enunciado:

- ***Jogador*** -> refere-se sempre ao ***personagem no jogo*** (o guarda redes, um avançado, etc., controlado pelo computador ou por um ser humano.
- ***Utilizador*** -> refere-se ao ***ser humano*** que está a usar este simulador, controlando um jogador de uma das equipas.

Haverá duas equipas. Em cada equipa haverá três tipos de ***jogadores*** apenas: guarda-redes, defesa, avançado, tendo estes jogadores características diferentes. Todos os jogadores terão movimento, sendo cada um controlado ou pelo computador, ou por um jogador. Pode haver vários ***utilizadores*** e cada um controla o jogador que quiser da equipa que entender. Os vários utilizadores podem pertencer à mesma equipa ou não, conforme lhes apetecer.

O campo de jogo consistirá num rectângulo de 21x51 posições em que cada posição no campo corresponde a uma posição do ecrã em modo texto. Os jogadores serão representados por letras/caracteres. A bola será também representada por uma letra/caracter.

O simulador vai envolver vários programas, nomeadamente um que age como **servidor** e controla todos os aspectos do jogo mas pouco ou nada interage com o utilizador/jogador, e outros que serão os **clientes**, que são essencialmente a interface entre os utilizadores e o servidor.

Poderá haver entre 0 e N utilizadores em simultâneo, sendo N o número total de jogadores nas duas equipas. Um jogador estará a ser controlado por um utilizador ou pelo computador. O utilizador usa um programa

cliente que recolherá a intenção do utilizador (teclas) e enviará essa informação ao servidor. O servidor envia a todos os clientes o estado actual do jogo e valida todas as acções solicitadas pelos clientes. É importante salientar que a interacção entre o servidor e o cliente não é o modelo diálogo: o servidor pode enviar dados ao cliente sem este ter enviado nada ao servidor. Tanto o servidor como o cliente deverão estar preparados para atendar várias fontes de informação em simultâneo (exemplo: o cliente tem que atender tanto as comunicações do servidor como as interacções como utilizador).

Matéria envolvida:

- O essencial do trabalho prático tem a ver com a matéria de SO: mecanismos de comunicação (*named pipes*), sinais, gestão de processos, *threads*, mecanismo *select*, etc.
- A implementação do jogo vai necessitar de funções da biblioteca *ncurses* que possibilitam o posicionamento do cursor em coordenadas *(linha, coluna)*, permitem mudar a cor do texto, etc., e que se usam de forma semelhante aos "printfs". O uso dessas funções não tem nada de especial mas será fornecido um pequeno exemplo.

Nota: As regras e condições de "fim de jogo", "ganhou jogo", perdeu jogo" são de menor relevância para o trabalho. Contar os golos marcados pela equipa é o suficiente.

2. Arquitectura geral

O jogo é obrigatoriamente constituído por vários processos a correr na mesma máquina Unix, interagindo entre si como cliente-servidor. Estes processos executam um de dois programas: "servidor" ou "cliente".

- **Servidor**

  Este programa armazena e gere toda a informação relativa ao jogo, e controla todos os aspectos do jogo. Só deve estar a correr um processo com este programa de cada vez.

  A interacção com o utilizador deste programa não tem a ver com o controlo de um jogador mas sim com o controlo do jogo propriamente dito: haverá uma espécie de "administrador" que configura/controla o jogo (ex., gestão de utilizadores e do jogo que está a decorrer). A interacção com o este utilizador é feita através do paradigma da *linha de comandos* (comandos escritos por extenso).

  O servidor controla todos os jogadores (personagens do jogo) que não estejam a ser controlados por utilizadores. Cada um destes jogadores controlados pelo computador corresponderá a uma *thread*.

  O servidor mantém uma lista de utilizadores autorizados a jogar, armazenados num ficheiro de texto, um utilizador por linha, no formato "username password"

- **Cliente**

  Este programa serve de interface do sistema com os utilizadores. É neste programa que é apresentado o campo do jogo e feita a interacção com os utilizadores. O programa tem como missão interagir com o utilizador e **não valida regras nem armazena informação local** acerca do jogo. Cada utilizador executa um cliente. O servidor só aceita comandos do utilizador depois de este ter fornecido uma autenticação login/password válida. O reconhecimento mútuo servidor-cliente é independente desta

autenticação e é estabelecido logo que o cliente se inicia. A autenticação pode ser pedida ao utilizador e enviada ao servidor a qualquer altura, mas só depois desse passo é que o servidor cumpre os comandos recebidos desse cliente.

Os utilizadores podem juntar-se a um jogo que esteja a decorrer a qualquer instante, mesmo eu o jogo já vá a meio, escolhendo a equipa e o jogador (personagem do jogo) que querem controlar, desde que este esteja "livre" (não esteja a ser controlado por outro utilizador). O controlo do jogador é depois feito com teclas definidas pelos alunos.

O processo "servidor" interage com os processos "cliente" apenas através de *named pipes* e *sinais*. Os processos "clientes" não interagem diretamente uns com os outros. Qualquer informação que seja necessário transferir de um "cliente" para o outro passará sempre pelo servidor, que mediará e reencaminhará essa informação.

Todos os programas são lançados através da linha de comandos. Cada programa deverá ser lançado a partir de uma sessão/*Shell*/*terminal* diferente, mas utilizando a mesma conta de utilizador na máquina.

3. Requisitos e descrição adicional

Servidor

- Apenas deve existir um processo "servidor" em execução. Caso seja executada uma nova instância do "servidor", esta deverá detectar a existência da primeira e terminar de imediato.
- Quando lançado, o "servidor" recebe através da linha de comandos o nome de um ficheiro de texto que contém o *username* e a *password* dos utilizadores. Este ficheiro é lido e escrito pelo "servidor". O servidor pode acrescentar utilizadores e o ficheiro pode estar inicialmente vazio.
- Quando termina, o "servidor" deve informar todos os "clientes" do sucedido.
- O "servidor" deverá terminar caso receba o sinal SIGUSR1.
- Os jogadores que não são controlados por utilizadores são controlados por *threads*. É obrigatório o uso deste mecanismo para este efeito. Se um utilizador abandonar o jogo, o jogador por ele controlado volta a ser controlado por uma *thread* do computador. O movimento dos jogadores quando controlados pelo computador é descrito mais adiante.
- Não é possível ter dois jogadores na mesma posição. O acesso das várias *threads* aos dados deve ser devidamente acautelado através dos mecanismos adequados para evitar incoerência nos dados.
- O "servidor" pode ser administrado por um utilizador (administrador) através de comandos escritos (identificados mais adiante).
- O "servidor" poderá ter que dar atenção a mais do que uma tarefa e entrada de dados em simultâneo (por exemplo: ler *pipes* e teclado). Esta situação deve ser resolvida de forma elegante usando os mecanismos estudados.

Cliente

- Podem estar a correr vários processos "cliente" em simultâneo – um por utilizador.

- Inicialmente o "cliente" contacta o "servidor" para que este fique a saber que existe. Posteriormente, o cliente informa que se pretende juntar a um jogo, especificando qual o jogador/equipa que deseja controlar e o servidor validará que essa escolha é possível.
- A qualquer instante o jogador poderá abandonar o jogo ou mesmo abandonar o programa de todo. O servidor deverá ser informado e reagir correctamente.
- A interface do cliente é totalmente em "modo texto". A apresentação do mapa, jogadores, bola, etc. ultrapassa a funcionalidade das bibliotecas C *standard*. **A construção da interface do cliente será feita com recurso à biblioteca *ncurses*.** Esta biblioteca não tem nada de especial mas será fornecida numa aula um exemplo sobre esta biblioteca.
- O cliente manterá uma visão do jogo permanentemente actualizada: mesmo que o utilizador desse cliente não faça nada, outros utilizadores poderão estar a fazer (ou o computador ao controlar os seus personagens), e o cliente manterá o seu utilizador a par. Este aspecto 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 uma tarefa /entrada de dados em simultâneo (por exemplo, lidar com informações vindas do servidor, teclado etc.). Esta situação deve ser resolvida de forma elegante usando os mecanismos apropriados e estudados.

4. Lógica de funcionamento

4.1 Constituição e controlo do jogo

O Campo é constituído por um rectângulo na horizontal de 21x51 posições. As balizas estão nos extremos, ocupando 9 posições na vertical, centradas verticalmente.

A bola estará inicialmente no centro do campo.

O jogo é composto por duas equipas. Em cada equipa haverá sempre os jogadores:

- Um guarda-redes
- N defesas (entre 1 e 4). O número exacto de defesas é especificado na variável de ambiente NDEFESAS. Se a variável não existir, assume-se o valor 2.
- N avançados (entre 1 e 4). O número exacto de avançados é especificado na variável de ambiente NAVANCADOS. Se a variável não existir, assume-se o valor 2.

Os jogadores podem deslocar-se na horizontal, vertical ou diagonal, uma posição de cada vez. Deslocam-se a uma velocidade que é de N em N décimas de segundo. O movimento é controlado da seguinte forma: o cliente/utilizador usa teclas para indicar ao servidor qual a direcção que deve fazer avançar o jogador. O servidor efectua esse movimento (se for possível) e só deixa fazer novo movimento para esse jogador passados as N décimas de segundo que lhe estão associados: guarda-redes, defesa e avançado têm velocidades diferentes.

Os jogadores têm um grau de precisão de remate. Rematar significa passar a bola a outro jogador ou à baliza oposta (ou seja, na direcção aproximada em que estes se encontram). Se o remate for bem sucedido, a bola vai na direção desejada. Caso contrário, vai numa direcção aleatória (ver mais adiante: movimento da bola).

Propriedades dos jogadores:

- O guarda-redes está inicialmente à frente da sua baliza. Pode deslocar-se apenas na extremidade do campo (até 7 posições de distância da sua extremidade do campo). Tem uma velocidade de "3 em 3" décimas de segundo. Tem uma precisão de remate de 25%.
- Os defesas deslocam-se em qualquer zona do campo. Têm a velocidade de "4 em 4" (são mais lentos) e um grau de precisão de 80%.
- Os avançados deslocam-se em qualquer zona do campo. Têm a velocidade de "3 em 3" e um grau de precisão de 60%.

Posse da bola:

- Os jogadores adquirem a posse da bola quando atingem uma posição adjacente a ela. Quando na posse da bola, o movimento do jogador propaga-se à bola (se o jogador andar para a esquerda, a bola também, etc.).
- Um jogador que passe ao lado da bola (posição adjacente) fica na sua posse mesmo que ela estivesse na posse de outro jogador (quem passa por último adquire-a).
- A bola é largada quanto algum outro jogador a "rouba", ou quando é rematada.

Movimento da bola:

- A bola desloca-se à velocidade do jogador que a tem, ou do jogador que a rematou.
- Não há atrito neste jogo e a bola só pára quando atinge o limite do campo, ou é apanhada por um jogador.
- Sempre que embate numa baliza é contabilizado um golo e passa para a posse do guarda-redes dessa baliza (implica que a bola é colocada junto dele).
- A bola anda na horizontal, vertical ou diagonal, uma posição de cada vez, à velocidade do último jogador que a teve.

Remate da bola:

- A bola pode ser rematada para outro jogador (exemplo: o utilizador prime a tecla que identifica o jogador destino: 0: guarda redes, 1 a 4 defesa da sua equipa, 5 baliza oposta, 6 a 9 avançado da sua equipa). Se o remate for bem-sucedido (precisão de remate = % ou probabilidade de sucesso), a bola irá na direcção pretendida. Caso contrário, irá numa direcção aleatória (mais pormenores sobre o movimento da bola a seguir). O servidor permitirá que a bola seja rematada para um jogador da outra equipa: é ao utilizador que compete estar com atenção.

- Remate em direcção a um, jogador ou baliza:
    - A bola irá na horizontal/vertical/diagonal na direcção aproximada do destino. A direcção da bola pode não coincidir exactamente devido à restrição horizontal/vertical/diagonal, e este facto tema ver como uma simplificação de algoritmos.

      Exemplo:
    - Bola rematada para jogador x. Jogador x está à direita (então a bola vai deslocar-se para a direita) e abaixo (então a bola vai deslocar para a direita + para baixo, resultando no movimento diagonal direita+baixo).
- Remate aleatório (precisão do jogador falhou):
    - O computador sorteia uma direcção dentro das possíveis (horizontal/vertical/diagonal).

Aspectos visuais:

- Cada equipa terá os seus jogadores em cores diferentes.
- A bola é representada por um "o", com a cor da equipa do último jogador que a teve/tem.
- Os jogadores são representados por números:
    - 0 = guarda redes
    - 1 – 4 -> defesas
    - 6 – 9 -> avançados

Fim de jogo

- O jogo termina quando se esgota o tempo estabelecido pelo servidor, ou quando o administrador decide terminar o jogo por comando. O servidor e os clientes continuam a funcionar, os clientes continuam *logados* e podem juntar-se a outro jogo assim que este comece.
- Quando o jogo termina todos os clientes são avisados e recebem informação acerca de quem ganhou.

4.2 Ligação ao servidor e ao jogo

Quando lançados, os clientes pedem o *username* e *password* ao utilizador e identificam-se junto do servidor. O servidor fica a saber que os clientes existem, mas estes ainda não estão ligados ao jogo. Para se ligarem ao jogo (ligar ao servidor ≠ ligar a um jogo), o cliente terá que activar a funcionalidade/opção de ligar ao jogo, escolhendo o jogador/equipa

- Quando um jogo termina, os utilizadores envolvidos voltam à situação de "não estão ligados a nenhum jogo", mas continuam associados ao servidor e este continua a saber que esses clientes existem.
- Se um utilizador que controla um jogador abandonar o jogo, esse jogador passa a ser controlado novamente pelo servidor (uma *thread* dedicada a esse jogador).

A única forma de um utilizador mudar de identidade (assumir outro *username*) será sair e voltar a ligar ao jogo. O utilizador pode assumir o controlo de outro jogador da mesma equipa (largando o jogador que estava a

controlar nesse instante) desde que o jogador alvo não esteja a ser controlado por nenhum outro utilizador nesse momento. Para o efeito o utilizador usa/selecciona uma letra/tecla que identifique esse jogador (teclas escolhidas por cada grupo, sugestão: os dígitos que identificam os jogadores).

4.3 Criação/terminação de um jogo

Um novo jogo tem início quando o administrador do servidor escreve o comando "start *n*" em que n é o número de segundos de duração do jogo. O jogo decore durante o tempo definido nesse comando, podendo ser interrompido a qualquer altura pelo comando stopr". Os utilizadores podem ligar-se a um jogo a decorrer em qualquer instante – não há prazo. Quando um jogo termina, os clientes são informados do resultado. Os clientes continuam ligados ao servidor (e continuam "autenticados") e podem ligar-se a outro jogo que entretanto comece.

4.4 Controlo do jogo no servidor

O servidor pode ser controlado directamente por um administrador através de uma interface semelhante a uma linha de comandos. Os comandos que o servidor reconhece são os seguintes:

- **start *n***

  Começa um novo jogo (se não estiver nenhuma a decorrer). Os clientes *logados* são avisados.
- **stop**

  Termina o jogo que estiver a decorrer. Os clientes são avisados e informados acerc de quem ganhou.
- **user *username password***

  Cria uma conta nova com o *username* e *password* especificados a não ser que já exista uma conta com esse *username*. A informação relativa ao novo utilizador deve ser armazenada em ficheiro.
- **users**

  Mostra a lista de utilizadores activos (*logados* no sistema/em jogo).
- **result**

  Mostra o resultado actual do jogo que estiver a decorrer.
- **red *username***

  Corresponde a uma espécie de cartão vermelho: o utilizador sai do jogo e o personagem (jogador) que ele estava a controlar reverte ao controlo do computador. O utilizador pode voltar a entrar no jogo.
- **shutdown**

  Desliga o servidor. O servidor deve terminar o jogo e informar os clientes desse facto. Toda a informação relevante deve ser armazenada em disco e os recursos temporários utilizados eliminados (não se trata de fazer "savegames").

A um dado instante podem estar vários clientes a correr. É admissível que seja posto um limite para simplificação das estruturas de dados no servidor (por exemplo, 20 clientes).

4.5 Controlo de jogadores pelo utilizador

A interface com o utilizador (no cliente) deve ser clara e racional. Não se vai micro-gerir o cliente, sendo dados apenas os requisitos gerais:

- Deve ser possível ver o campo, distinguindo os jogadores por tipo e por equipa.
- O campo deve estar sempre actualizado (sempre que há uma alteração no servidor o cliente é informado). Devem ser visíveis no ecrã os dados sobre o jogo (pontuação, tempo, número de utilizadores em campo, etc.)
- O utilizador deve poder
    - Logar-se/sair, ligar a um jogo (escolhendo jogador), sair de um jogo (lógica de comandos escritos)
    - Movimentar o seu jogador e mudar para outro jogador na equipa (lógica de teclas direcção/outras)

4.6 Controlo de jogadores pelo computador

Não é necessário inteligência artificial. Basta que os jogadores se desloquem em direcção à bola, acrescentando coisas tais como: de vez em quanto (aleatório) tenta rematar à baliza ou passar a outro jogador da equipa. Também se pode ter os defesas a vaguearem perto da sua baliza em vez de irem a correr feito totos para bola (deixando a baliza desprotegida). De igual forma pode-se ter os avançados a terem tendência a irem sempre atrás da bola onde quer que ela se encontre (ou seja, deslocarem-se na direcção aproximada dela). Seja criativo sem no entanto gastar demasiado tempo neste aspecto secundário.

5. Extras opcionais

O trabalho contempla um conjunto de extras que podem elevar a nota final do mesmo (até ao máximo de 100%), compensando partes do trabalho menos bem conseguidas. Os extras são:

- "Inteligência" artificial nos jogadores controlados pelo computador (10%)
- Solução gráfica com Qt (15%)
- Elaboração de *makefile* para o projecto, com lógica devidamente explicada no relatório (5%)

Os extras não estão abrangidos por matéria em aulas, sendo por isso mesmo *opcionais* e valorizados como *extra*. Podem ser feitos vários extras em simultâneo, mas nunca ultrapassando 15%.

6. Considerações adicionais

- Podem existir diversas situações de erros potenciais que não são explicitamente mencionadas no enunciado. Estas situações devem ser identificadas e o programa deve lidar com elas de uma forma controlada e estável. O terminar abruptamente o programa não é considerado uma forma adequada.

- O enunciado não pretende fazer *micro-gestão* dos alunos. É natural que haja aspectos em aberto que devem ser identificados e resolvidos por bom senso, tal como acontece na indústria.
- Caso existam *bugs* no enunciado, serão corrigidos e a sua correcção anunciada nas aulas ou no moodle. Poderá haver lugar a documentos adicionais no *moodle*, nomeadamente um *Frequently Asked Questions* com as perguntas mais frequentes e respetivas respostas.

7. Regras gerais do trabalho

Aplicam-se as seguintes regras:

- O trabalho pode ser realizado em grupos de um ou de dois alunos (sem exceções). No caso de ser realizado individualmente, o trabalho (e a valorização) é o mesmo.
- Este trabalho não são dois subtrabalhos: Não se aceita que um grupo divida o trabalho em duas metades, sendo cada metade realizada exclusivamente por um dos alunos. Ambos os elementos do grupo devem trabalhar no todo e ser capazes de responder e defender o trabalho todo.
- O trabalho vale cinco valores na nota final da disciplina. Tem mínimos de 25% que serão dispensados para os alunos que tenham 10 presenças nas aulas práticas.
- Prazo de entrega: **8 / janeiro / 2017**. Cada dia de atraso na entrega será penalizado com 20% de desconto. Não se prevê que seja possível aumentar o prazo.
- A entrega do trabalho é feita sob a forma de um arquivo ***zip*** (leia-se "***zip***" - não é *arj, rar, tar,* ou outros) contendo os ficheiros do projeto (código fonte) e o PDF do relatório. O arquivo tem obrigatoriamente o nome

  so_1617_tp_nome1_numero1_nome2_numero2.zip

  em que **nome1** e **numero1** são o nome e número de um dos alunos do grupo e **nome2** e **numero2** dizem respeito ao segundo aluno do grupo (se existir).
- Não existem metas intermédias.

8. Avaliação do trabalho

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

- **Arquitectura do sistema** – Há aspectos relativos à interacção 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 haver mal entendidos.
- **Implementação** – Deve ser racional, não desperdiçar recursos do sistema. As soluções encontradas para cada problema do trabalho devem ser claras. O estilo de programação deve seguir as boas práticas de programação. O código deve ter comentários relevantes. Os recursos do sistema devem ser usados de acordo com a sua natureza.

- **Relatório** – Deverá ser escrito e entregue um relatório completo descrevendo a estratégia e modelos seguidos, a estrutura da implementação e as opções tomadas (máximo de 10 páginas). Este relatório será enviado juntamente com o código no arquivo submetido via *moodle* e também será 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 na sua realização.

  Nota: 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.
- 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.