Trabalho Prático

Programação em C para Unix

Exemplo de jogo MUD – *Multi-User Dungeon*

Este documento apresenta um exemplo de um jogo MUD hipotético para melhor contextualizar aquilo que está a ser pedido no enunciado do trabalho. Trata-se apenas de um exemplo, e aquilo que é efectivamente pedido no trabalho encontra-se no enunciado e não aqui.

1. Jogos MUD

Os jogos MUD consistem em labirintos (*dungeons*) nos quais os jogadores navegam. O labirinto é constituído por salas interligadas umas às outras através de passagens. As salas podem ser de diferentes tipos (cela, sala, corredor, etc.) e ter diversas propriedades que poderão ou não afectar o jogador (uma sala escura, que dificulta o movimento, ou a defesa, uma sala com chão muito frio, que pode roubar saúde, etc.). As salas podem conter objectos, de diversos tipos, que poderão afectar o jogo (exemplo, comida, armas, etc.). Poderão existir diversos inimigos (geralmente monstros, em diversas variedades) no labirinto, e, normalmente, quando um jogador entra numa sala onde existe um monstro, é quase certa uma batalha. O labirinto pode ser pequeno, ou arbitrariamente grande, tornando o jogo mais interessante. Os jogos MUD podem ser muito ricos em termos de conteúdo, mas isso não significa que o trabalho de SO tenha que ter essa variedade e complexidade.

Geralmente existe um objectivo no jogo: apanhar um determinado objecto, salvar um prisioneiro, fazer um duelo com outro jogador, ou simplesmente explorar e apanhar objectos valiosos pelo caminho. Uma parte importante do jogo é o factor aleatório: o labirinto pode ser gerado aleatoriamente, uma batalha pode ser decidida com base na força do jogador, do monstro, das armas usadas, e também em factores aleatórios, tornando o jogo sempre diferente de cada vez que se joga. O trabalho de Sistemas Operativos não será necessariamente um MUD ao nível de complexidade dos jogos reais.

A interface com o utilizador destes jogos é do tipo consola. A informação dada ao jogador acerca de tudo o que se passa no jogo é constituída por simples mensagens escritas no ecrã. As acções do jogador são dadas ao computador através de comandos: *strings* que serão analisadas e processadas pelo jogo. Não existe e nem é necessário gráficos, rato, ou qualquer conteúdo multimédia.

Como exemplo de interacção como jogador apresenta-se a seguinte situação a meio de um jogo. Todo o texto foi apresentado pelo jogo excepto aquele que começa com um ">", que são os comandos introduzidos pelo jogador.

```
Entraste numa sala escura. A porta fechou-se automaticamente atrás de ti.
À frente existe uma porta. No meio da sala encontra-se uma aranha gigante de aspecto ameaçador. Podes tentar passar por ela, ou atacar.
O que pretendes fazer?
> mover frente
Moves para frente.
A aranha apanhou-te de surpresa e envenenou-te. Morreste.
```

Neste exemplo de situação (e trata-se apenas de um exemplo, não significa que seja necessário implementar aranhas gigantes no trabalho), há que notar os seguintes aspectos.

1. Para cada acção do jogador, acontece sempre algo. O jogo apresenta o que acontece sempre após cada acção desencadeada pelo jogador.
2. Houve uma acção do jogador (não representado no exemplo) que levou a que o jogador entrasse numa sala. O jogo apresenta a nova situação, dando os detalhes ("entraste numa sala escura etc.").
3. Após apresentar a nova situação em que o jogador se encontra ("estás na sala com uma aranha e uma porta, etc."), o jogo pergunta o que o jogador pretende fazer. O tempo pára enquanto o jogador não responde. Não há lugar a questões tais como "a aranha vai-me apanhar, tenho que responder depressa", pois o jogo só avança depois do jogador responder.
4. O jogador indica a sua intenção através de um comando. Neste caso "mover frente". O comando é "mover" e a direcção é indicada com uma palavra adicional, semelhante ao conceito da linha de comandos e argumentos de comandos[1]. As palavras que o jogador escreve seguem uma lógica (sintaxe) muito simples. Não se pretende de forma alguma que o jogo entenda linguagem natural (Português, Inglês, ou outra que fosse).
5. Os comandos introduzidos pelo jogador são validados pelo jogador e, eventualmente, alguns dos comandos podem ser rejeitados por não se aplicarem na situação actual do jogo (por exemplo, tentar "descer" quando não existem escadas na sala será simplesmente ignorado).
6. O jogo executa a acção expressa nos comandos introduzidos pelo jogador, passando o jogo para uma nova situação. Assim sendo, o jogo apresenta o resultado (por exemplo, "entraste num corredor comprido. Há um tesouro no fundo"). Neste caso, o resultado foi tentar passar pela aranha e a coisa não correu bem para o lado do jogador.

No exemplo seguinte, vai-se explorar o que poderia acontecer se o jogador tivesse decidido fazer outra coisa em vez de passar pela aranha (também poderia fugir, mas a porta por onde entrou fechou-se atrás dele[2]).

[1] Note-se que estamos no meio da execução de um programa. O paradigma é semelhando ao da linha de comandos, mas neste momento não se está no contexto de Shell, mas sim de um programa que por acaso está a efectuar vários scanf/gets/etc.
[2] Isto não significa que o trabalho de SO tenha que implementar portas automáticas.

```
Entraste numa sala escura. A porta fechou-se automaticamente atrás de ti.
À frente existe uma porta. No meio da sala encontra-se uma aranha gigante de aspecto ameaçador. Podes tentar passar por ela, ou atacar.
O que pretendes fazer?
> ver aranha
A primeira vista, parece ter força de ataque 5, defesa 3, saúde 10.
> info
Tens saúde 15
Carregas 4 objectos
  Faca – força de ataque 8
  Escudo de cartão prensado – defesa 3
  Fisga – força de ataque 1
  Sandes de atum – recupera 3 pontos de saúde
O que pretendes fazer?
> atacar aranha faca
Atacas a aranha. Desferes um golpe com faca. A aranha sofre dano 6. A aranha contra-ataca com força 5. Sofres dano 4. Desferes um golpe com faca com força 8. A aranha sofre dano 5. A aranha morre.
O que pretendes fazer?
> ver
Estás numa sala escura. À frente existe uma porta.
O que pretendes fazer?
>
```

Neste segundo exemplo, vê-se o resultado, no mesmo cenário de fundo, do que poderia acontecer se o jogador tivesse optado por atacar a aranha:

1. O jogador usa o comando "ver" para analisar melhor as propriedades da aranha. O jogo informa-o com alguns dados. Aqui, trata-se apenas da força de ataque e robustez da aranha.
2. O jogador olha para si próprio para ver o seu estado e comparando-se com a aranha para ver se tem boas hipóteses numa luta corpo a corpo com essa criatura. Para tal usa o comando "info". O jogo informa-o dos dados relevantes. Notar a existência de uma faca de força de ataque 8 no inventário do jogador.
3. Com base na informação acerca do inimigo, e de si próprio, o jogador decide atacar. Para tal usa o comando "atacar" especificando o alvo "aranha" e a arma "faca".
4. O jogo simula o ataque. Depois de iniciado, o ataque é até à morte de um dos intervenientes. Neste caso, o jogador, como tomou a iniciativa, desfere o primeiro golpe. A faca tem força 8, mas o jogo sorteou que o dano sofrido pela aranha é de apenas 6 (do máximo de 8), eventualmente porque a aranha se defendeu. A forma como estes valores são obtidos é especificada mais adiante – neste momento está apenas a ver-se a lógica deste tipo de jogos. A aranha não morreu logo e contra-ataca, infligindo um dano de 4 (quase o máximo da sua capacidade de ataque). O jogador desfere novo golpe arrumando a aranha. Fim de combate.
5. O jogador usa o comando "ver" para saber o ponto da situação. Reparar que já não há aranha nenhuma.

Para encerrar este exemplo, apresenta-se agora a situação em que o jogador usa um objecto. Este exemplo vem na sequência do combate anterior.

```
O que pretendes fazer?
> info
Tens saúde 11
Carregas 4 objectos
  Faca – força de ataque 8
  Escudo de cartão prensado – defesa 3
  Fisga – força de ataque 1
  Sandes-atum – recupera 3 pontos de saúde
O que pretendes fazer?
> usar sandes-atum
Usas a sandes-atum. Recuperas 3 pontos de saúde.
> info
Tens saúde 14
Carregas 3 objectos
Faca – força de ataque 8
Escudo de cartão prensado – defesa 3
Fisga – força de ataque 1
O que pretendes fazer?
>
```

Neste terceiro exemplo é de reparar no seguinte:

1. O jogador faz uma pequena introspecção acerca de si próprio usando o comando "info". A sua saúde agora é de apenas 11, pois sofreu um dano de 4 no decorrer do ataque à aranha.
2. O jogador decide usar um dos seus objectos. Para tal especifica o comando "usar" indicando o nome do objecto pretendido: "sandes-atum". O efeito do uso deste objecto consiste em duas coisas: a) a saúde aumenta 3 pontos, e b) o objecto desaparece (este objecto é de apenas um uso).
3. O jogador confirma, através do comando "info" que de facto está um pouco mais saudável (14 pontos de saúde) e que ficou sem a sandes de atum.

A próxima acção do jogador poderia agora ser "mover frente" para ver que tesouros maravilhosos (ou mais aranhas gigantes) existirão depois da porta, mas isso já não faz parte deste exemplo.

2. Considerações e simplificações no contexto de Sistemas Operativos

2.1 Acerca do texto escrito pelo jogo

No exemplo apresentado acima, o texto impresso pelo jogo é razoavelmente bom. Este grau de perfeccionismo exige alguns algoritmos que não estão no contexto do primeiro ano. Assim, é natural e aceite que o texto apresentado pelo jogo no trabalho de Sistemas Operativos tenha um aspecto mais macarrónico. Por exemplo, ao entrar numa nova sala, em vez do texto:

```
Entraste numa sala escura e fria. Existem duas portas, uma a norte, e outra a este. No meio está uma aranha gigante ameaçadora
```

Poderá aparecer apenas:

```
Sala escura e fria.
Portas: norte, este
Tem: aranha
```

Esta forma de construção de texto é de implementação directa. Partindo do pressuposto que existirá uma estrutura de dados qualquer que descreve o labirinto e o conteúdo de cada sala (por exemplo, uma matriz bidimensional de estruturas), então o algoritmo limitar-se-á a apresentar algum texto que está armazenado nessa estrutura de dados. Essa estrutura conterá, por exemplo, a descrição ("sala escura e fria"), as portas que existem (norte e este), e o conteúdo, quer de inimigos (um: aranha), quer de objectos (neste exemplo, nada).
Naturalmente, a simplificação de texto descrita aqui aplica-se a todo o jogo e não apenas à descrição das salas.

2.2 Acerca da complexidade de objectos e comportamentos

Quanto maior a riqueza e diversidade de objectos, personagens e acções, mais interessante este se tornará, mas também mais complexa a sua implementação será. No contexto da disciplina de Sistemas Operativos, o que está em questão não é a complexidade do argumento de um jogo, mas sim a correcta utilização de alguns dos mecanismos usados na sua implementação. Assim, o trabalho prático incide numa versão simplificada deste tipo de jogos, com um conjunto reduzido de objectos, personagens e acções, os quais são enumerados no enunciado propriamente dito. Portas com capacidade de se fecharem sozinhas, objectos armadilhados, e outras coisas interessantes, ficarão para uma possível continuação do trabalho num contexto que não o de sistemas operativos.

2.3 Acerca das estruturas de dados a usar

Apesar do aspecto versátil que estes jogos por vezes aparentam (com grande variedade de diálogo com o utilizador, e grande variedade de acções possíveis), as estruturas de dados a usar são relativamente simples. Por exemplo, se se assumir um labirinto de dimensões fixas, pode-se usar uma simples estrutura para descrever as propriedades de uma sala, e uma matriz bidimensional (em que cada elemento é do tipo dessa estrutura) para descrever a totalidade do labirinto. A posição dos jogadores e dos monstros poderá ser mantida em variáveis à parte dessa estrutura, ou na própria matriz que descreve o labirinto.

2.4 Acerca da matéria de sistemas operativos

A parte do trabalho mais directamente relacionada com sistemas operativos tem a ver com a arquitectura da implementação e mecanismos do sistema usados. O jogo vai ser implementado por um processo central que gere toda a informação. A interface com cada jogador vai ser implementada através de um ou mais processos independentes. Os vários processos irão agir em conjunto, comunicando entre eles através de **mecanismos de comunicação Unix**, a ver nas aulas. A coordenação entre os processos passará pela utilização de **sinais** (outro mecanismos Unix a ver nas aulas). De igual forma, outros aspectos da implementação irão requerer mecanismos específicos do sistema Unix.