Trabalho de Laboratório
Programação de Sistemas Computacionais - LEE
IST - 2024/2025

Jogo Master Mind

1 Introdução

Este trabalho tem como objectivo fundamental a familiarização por parte dos alunos com os aspectos de concorrência, sincronização e comunicação oferecidos por um sistema operativo (Unix/Linux). Os aspectos da interface aplicacional com o utilizador são relegados para segundo plano, sendo portanto simplificados.

A aplicação que se pretende concretizar – um "Jogo Master Mind" – envolve os aspectos de comunicação entre o(s) utilizador(es) / jogador(es) e um servidor (constituido por diversos processos), bem como aspectos de comunicação e sincronização entre os diversos componentes do sistema, e ainda o registo e consulta de um histórico relativo à utilização da aplicação (interacção com o sistema de ficheiros).

2 Descrição geral do problema

A aplicação a programar (Jogo Master Mind (JMM)) é constituída por vários processos que comunicam entre si, e está dividida em três partes fundamentais: a interface com o utilizador/jogador ("`JMMapl`"), o controlo do jogo propriamente dito (servidor multitarefa para suportar vários jogadores) ("`JMMserv`"), e o registo histórico ("`JMMlog`") (ver Figura 1).

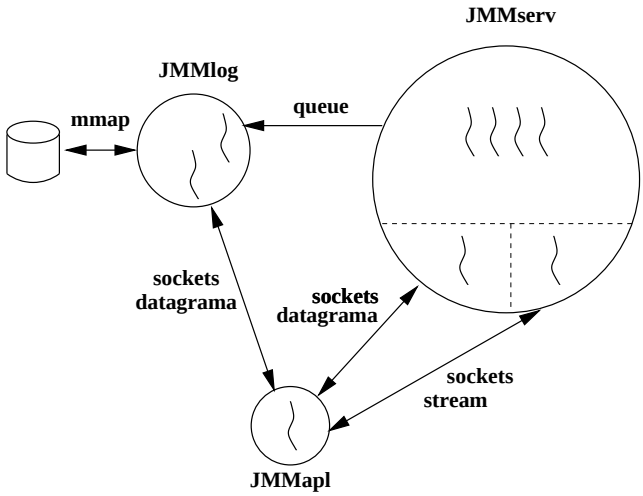

Figure 1: Arquitectura global da aplicação

A interface com o jogador ("`JMMapl`"), para além de permitir realizar o jogo propriamente dito, permite também efectuar comandos de consulta e configuração do sistema. Deverá ser garantida a possibilidade de estarem em execução simultânea mais do que um programa "`JMMapl`", ou seja, vários jogadores em simultâneo.

O servidor de jogo ("`JMMserv`") tem uma arquitectura multitarefa, sendo criada uma nova entidade concorrente para tratar de um novo jogo. Poderão existir até um máximo de `NJMAX` jogadores activos num determinado momento. Quando um jogo termina, os dados relativos a esse jogo podem ser

guardados de forma persistente, através do seu envio para o processo que faz o registo histórico (se esse envio estiver activo), para formar tabelas de classificação.

O registo histórico (`JMMlog`) recebe a informação proveniente do servidor, processa-a, e actualiza o ficheiro correspondente. Para além disso aceita comandos, da parte do jogador, para consulta e gestão da informação guardada.

Para que a aplicação global tenha a funcionalidade completa é necessário que os programas `JMMserv` e `JMMlog` estejam em execução. No entanto, deve ser possível suportar o funcionamento parcial de forma independente, ou seja, ter só a parte relativa ao `JMMserv` ou só a parte relativa ao `JMMlog`.

Para se poderem terminar de forma ordeira, e salvaguardando toda a informação relevante, tanto o `JMMserv` como o `JMMlog` devem aceitar uma ordem de terminação, que pode ser enviada a partir da aplicação **JMMapl**, e também através de um sinal **SIGTERM** enviado a partir da linha de comando ("shell").

3 Interface com o jogador – JMMapl

O programa responsável pela interface com o jogador (`JMMapl`) suporta um conjunto de comandos para interagir quer com o servidor do jogo propriamente dito (`JMMserv`), quer com o processo de registo histórico (`JMMlog`). São os seguintes, os comandos a disponibilizar:

Comandos Disponíveis		
comando	argumentos	descrição
cnj	N n	- começar novo jogo (Nome (3 chars) e nível de dificuldade (1/2))
jg	$ABC[DE]$	- jogada (3 ou 5 letras, dependendo do nível)
clm		- consultar limites (número máximo de jogadas, tempo limite)
mlm	j t	- mudar limites (número máximo de jogadas, tempo limite (minutos))
cer		- consultar estado envio de registos para histórico
aer		- activar envio de registos para histórico
der		- desactivar envio de registos para histórico
tmm		- terminar processo master mind (JMMserv)
ltc	n	- listar tabela(s) classificação nível n (0-todos)
rtc	n	- reinicializar tabela(s) classificação nível n (0-todos)
trh		- terminar processo de registo histórico (JMMlog)
sos		- mostrar comandos disponíveis
help		
sair		- sair da aplicação de jogo (JMMapl)

No primeiro grupo de comandos a interacção é feita com o servidor (`JMMserv`), e no segundo grupo de comandos a interacção é feita com o processo de registo histórico (`JMMlog`). Em qualquer dos casos existe uma mensagem de resposta, ficando o processo bloqueado à sua espera (interface síncrona). Deve, no entanto, ser prevista a hipótese de impossibilidade de comunicação, com o correspondente desbloqueio ("timeout"). No caso em que o comando especificado não seja executado com sucesso remotamente, a mensagem de resposta terá um código de erro.

4 Servidor de Jogo Master Mind – JMMserv

Como se disse atrás, o servidor de jogo ("`JMMserv`") possui uma arquitectura multitarefa, existindo vários "fios-de-execução" ("threads"). Para além da "thread" inicial, que permite uma interação

genérica com os utilizadores para execução dos comandos referidos anteriormente, existe uma "thread" responsável pela aceitação de pedidos de um novo jogo, e uma "thread" por cada jogo activo, criada na sequência da aceitação do pedido de jogo, e que durará enquanto durar esse jogo.

As várias "threads" de jogo são idênticas entre si (mesmo código), distinguindo-se apenas pelos parâmetros que lhe estão associados. Podem existir, num determinado momento, até um máximo de `NJMAX` "threads" de jogo. Cada "thread" de jogo dialogará com o respectivo jogador enquanto durar o jogo.

Aquando do início de um jogo será gerada de forma aleatória (com recurso à função `random`) a "chave secreta" desse jogo. Nesta "versão" de Master Mind, os pinos de diversas cores serão substituídos por uma sequência de letras, cujo comprimento e composição será dependente do nível de dificuldade seleccionado. No nível 1: 3 de 5 [ABCDE] (com possíveis repetições). Exemplo: ABA. No nível 2: 5 de 8 [ABCDEFGH] (com possíveis repetições). Exemplo: CHEFE.

Sempre que é recebida uma nova tentativa da parte do jogador, para além de se incrementar o contador de tentativas, será verificado o resultado e enviada a respectiva resposta, composta (nesta versão do jogo) por 2 números, correspondentes aos pinos pretos e brancos da versão original (número de cores/letras certas no sítio certo, número de cores/letras certas no sítio errado).

Quando um jogo termina com sucesso, e a opção de envio de registos está activa, será criado um registo de jogo (ver registo `rjg_t`) e enviado para o processo de registo histórico (`JMMlog`).

Deverá ser possível considerar limites, quer no número máximo de tentativas, quer no tempo máximo utilizado, casos em que o jogo terminará sem sucesso (valores iniciais `MAXNJ` e `MAXT`).

O programa `JMMserv` é activado de forma independente, sendo responsável pelas inicializações necessárias ao correcto funcionamento da aplicação (criação dos vários objectos de comunicação e sincronização). Caso o processo de registo histórico não esteja ainda disponível, e seja activado o envio de registos, também deverá ser criado a partir deste programa.

5 O registo histórico – JMMlog

O programa `JMMlog` pode ser activado de forma independente, ou a partir do programa `JMMserv`. É responsável por guardar em memória persistente (ficheiro **JMMLOG**) as tabelas de classificação construídas com base na informação recebida do servidor `JMMserv`, correspondente aos resultados de cada jogo. Para além disso, deve ainda permitir a consulta dessas tabelas feita a partir da interface do jogador (`JMMapl`). O ficheiro **JMMLOG** deve ser mantido entre as várias execuções de `JMMlog`.

Este programa possui 2 "fios-de-execução" ("threads"): um dedicado à comunicação com a interface do jogador (por exemplo o programa principal (main)); outro para receber a informação vinda do servidor de jogo `JMMserv`.

As tabelas a guardar no ficheiro correspondem aos melhores resultados, conseguidos até ao momento, em cada nível de jogo. Estarão ordenadas pelo número de tentativas usadas para obter a solução, e pelo tempo usado. Por cada nível serão mantidos os **TOPN** melhores resultados.

A gestão do ficheiro pode ser efectuada quer utilizando as primitivas mais tradicionais de acesso a ficheiros (`open, read, write, ...`), como mapeando o ficheiro em memória (`mmap`) (aconselhado).

6 Objectos de comunicação e sincronização

A sincronização entre as várias "threads" existentes num processo, como por exemplo o controlo do acesso destas a eventuais estruturas de dados partilhadas, deve ser feito utilizando semáforos POSIX (`sem_init, sem_post, sem_wait, ...`), ou as próprias primitivas associadas às "pthreads"

(`pthread_mutex_lock, pthread_mutex_unlock, ...`).

Na comunicação entre a interface do jogador (JMMapl) e o registo histórico (JMMlog) devem ser usados "sockets unix datagrama".

Na comunicação entre a interface do jogador (JMMapl) e o servidor de jogo (JMMserv) devem ser usados 2 tipos de comunicação: "sockets unix datagrama" para os comandos genéricos; e "sockets unix stream" para o jogo propriamente dito.

A comunicação entre as várias "threads" de jogo e o processo que faz o registo histórico deve ser efectuada por mensagens (*message queues*), utilizando a interface POSIX (`mq_open, mq_send, mq_receive, ...`). Esta comunicação com o processo de registo histórico tem uma interface assíncrona, não havendo qualquer mensagem de resposta.

São os seguintes os nomes globais associados aos referidos objectos de comunicação:

JMMSERVSD - Nome do socket unix datagrama do servidor `JMMserv`;

JMMSERVSS - Nome do socket unix stream do servidor `JMMserv`;

JMMLOGSD - Nome do socket unix datagrama do registo histórico `JMMlog`;

JMMLOGQ - Nome da queue POSIX do registo histórico `JMMlog`.

7 Desenvolvimento do projecto

No desenvolvimento do projecto, aconselha-se a utilização de uma estrutura modular, com testes faseados.

Na interface com o jogador, **não** se pretende nada de muito complexo (não é esse o objectivo fundamental). Para simplificar a sua concretização, os alunos **devem** utilizar o interpretador de comandos rudimentar (`cmd`), que é fornecido (ver página da disciplina).

8 Estruturas de dados e constantes

Na concretização do trabalho considere os seguintes valores para as constantes referidas anteriormente:

```
#define NJMAX       4                        /* número máximo de jogadores em simultâneo */
#define TOPN       10                      /* número de registos em cada tabela de nível */
#define MAXNJ      10           /* valor inicial do número máximo de jogadas (tentativas) */
#define MAXT        5                  /* valor inicial do tempo máximo de jogo (minutos) */
#define JMMLOG     "JOGOS.LOG"                          /* ficheiro com registo histórico */

#define JMMSERVSD  "/tmp/JMMSERVSD"         /* nome do servidor de jogo (socket datagram) */
#define JMMSERVSS  "/tmp/JMMSERVSS"           /* nome do servidor de jogo (socket stream) */
#define JMMLOGSD   "/tmp/JMMLOGS"          /* nome do registo histórico (socket datagram) */
#define JMMLOGQ    "/JMMLOGQ"                         /* nome do registo histórico (queue) */

typedef struct rjg_s {                                 /* estrutura de um registo de jogo */
  int nd;                                                 /* nível de dificuldade do jogo */
  char nj[4];                                           /* nome do jogador (3 caracteres) */
  int nt;                                                  /* número de tentativas usadas */
  time_t ti;                                          /* estampilha temporal início do jogo */
  time_t tf;                                             /* estampilha temporal fim do jogo */
} rjg_t;
```

9 Entrega e Visualização do Trabalho

Este trabalho tem uma **meta intercalar na semana de 17 de Março** (no horário de laboratório (5f-20/03 e 6f-21/03)). Deverão mostrar a funcionar, **pelo menos**, uma versão rudimentar do sistema que inclua:

- comunicação entre a interface do jogador `JMMapl` e o servidor de jogo `JMMserv` (comandos para consultar e modificar parâmetros, começar um novo jogo, terminar processo).

O **trabalho final** deve ser entregue até ao dia 6 de Abril de 2025. O material a entregar consiste numa cópia em formato digital (ficheiro ZIP) de todos os programas desenvolvidos e respectivo "makefile", e um pequeno relatório (3 ou 4 páginas) descrevendo as principais estruturas de dados usadas, bem como os aspectos relativos à comunicação e sincronização entre as várias entidades concorrentes existentes. Todos estes elementos devem ser identificados com o número do grupo de laboratório e alunos que o compõem.

As **visualizações e discussões dos trabalhos** serão efectuadas na **semana de 7 de Abril** (em horários a definir) com base no trabalho entregue.