SISTEMAS OPERATIVOS

PROJETO 2: IMPLEMENTAÇÃO DO *CONWAY'S GAME OF LIFE*

Realizado por:

M.ª Inês Coelho, 2004104311

Filipe Batista, 2006124931

CRIAÇÃO DO MUNDO

O projecto apresentado constitui uma implementação do *Conway's Game of Life. A* aplicação começa por pedir ao utilizador para definir a duração, em gerações, do jogo, o tamanho da grelha e a frequência de *snapshots*. Por questões de visualização optamos por limitar o tamanho da grelha entre 20 e 80 células. Posteriormente o utilizador pode definir as quantidades de cada padrão, entre os 5 disponíveis *(single, block, glider, LSS, pulsar),* que quer colocar em jogo. Para garantir a colocação aleatória dos padrões na grelha, são colocados primeiro os de maior dimensão e o espaço disponível para colocar novos padrões em jogo diminui progressivamente: até metade da grelha pode ser ocupada pelo primeiro padrão a ser colocado em jogo e os restantes padrões podem ocupar até um quarto do espaço livre da grelha. Estas funções encontram-se definidas em *pattern.c*.

GRANULARIDADE

Para obter uma melhor performance, optámos por utilizar uma granularidade linha a linha. O jogo consiste em duas funções: *evolve()* e *copy_grid()* (definidas em *game.c*). Cada execução destas funções processa uma linha da grelha: a função *evolve()* calcula a próxima geração de cada célula da linha para uma grelha temporária, que tem a mesma dimensão da grelha principal do jogo; a função *copy_grid()* copia uma dada linha da grelha temporária para a grelha principal.

MEMÓRIA PARTILHADA

Recorre-se a memória partilhada para permitir multi-processamento da grelha do jogo. Tanto a grelha principal do jogo como a grelha temporária são geradas numa área de memória partilhada. Também os semáforos e variáveis para armazenamento de posições de leitura e escrita são guardados numa estrutura do tipo *mem_struct* numa área de memória partilhada. No final da execução do programa, estas áreas de memória são libertadas. Todas estas funções podem ser encontradas no ficheiro *grid.c*.

MULTI-PROCESSAMENTO

A fução *main()* cria vários processos (o número de processos a gerar é definido numa macro em *header.h*) para computar a próxima geração da grelha. Os processos filhos vão executar a função *producer()*, enquanto o processo pai executa a função *consumer()* (funções definidas em *worker.c*).

Cada processo executa a função *producer()* até toda a grelha de jogo ter sido analisada durante as gerações correspondentes à duração do jogo. Em cada iteração da função, verifica-se a posição de leitura da grelha e

computa-se a geração seguinte dessa linha e incrementa-se o contador da posição. Este contador encontra-se armazenado numa área de memória partilhada e, para garantir que cada processo só executa uma linha, a sua leitura e processamento encontra-se guardada por um semáforo *mutex* designado de *mutex_grid*. Posteriormente, é incrementado o semáforo *cont* que funciona como um contador para o número de linhas cuja próxima geração já foi computada.

A função *consumer()* verifica se existem linhas cuja próxima geração já foi computada mas que ainda não foram copiadas para a grelha principal. Nesse caso decrementa o semáforo *cont*, requer acesso exclusivo à grelha principal e copia a próxima linha, cujo valor se encontra num contador armazenado em memória partilhada, da grelha temporária para a grelha principal do jogo.

SINCRONIZAÇÃO ENTRE PROCESSOS

A sincronização entre processos é conseguida através de semáforos, recorrendo a dois estratagemas:

(1) A função *producer()* não incrementa o contador do semáforo *cont* quando computa a primeira linha, sendo isso compensado no final da grelha, em que o contador é incrementado duas vezes. Desta forma, garantimos que a linha anterior à que está a ser computada não foi ainda substituída.

(2) Quando o número de linhas por copiar (isto é o valor do semáforo *cont*) excede metade da grelha, o semáforo *mutex_grid* executa um *wait* de 0,1s.

Desta forma, garantimos sincronização entre processos e entre a computação da próxima geração de uma dada linha e a sua cópia para a grelha principal do jogo. Verificou-se que, para uma grelha de grandes dimensões (80), povoada com pulsares e *blocks* (que se mantêm ao longo do tempo), para um elevado número de gerações (100.000), não houve perda de estrutura.

As operações com semáforos encontram-se definidas no ficheiro *semaphores.c.*

SNAPSHOTS

Para fazer os *snapshots* recorre-se a *memory mapped files*. Por cada *snapshot* é inicializado um novo ficheiro na pasta *snapshots* (que, se não existir no directório do programa, terá que ser criada), cujo nome correspondente é *AnoMêsDia_HoraMinuto_GeraçãoNúmero*, sendo que a parte do nome correspondente à data e hora é criada no início do programa para ser comum a todos os *snapshots* da mesma execução. As funções necessárias ao manuseamento dos *memory mapped files* encontram-se em *files.c.*

Dada a forma como o *Game of Life* foi implementado, todos os processos estão sincronizados, não sendo necessário o conceito de barreira. Quando uma geração é múltipla da frequência de *snapshots* definida pelo utilizador, é activada uma *flag* na função *consumer()* e, após uma linha ter sido copiada da grelha temporária para a grelha principal, essa mesma linha da grelha principal é copiada para a zona de memória partilhada e, sucessivamente, para o ficheiro *snapshot* correspondente. Ao atingir o final da grelha nessa geração, a área de memória é desmapeada e o ficheiro é fechado. Enquanto não se chegar à última geração do programa, inicializa-se o mapeamento de outra área de memória para um novo ficheiro, de modo a processar outro snapshot. A última geração é também copiada para um *snapshot*. O primeiro *snapshot* consiste na geração 0, ou seja, a grelha após a criação do mundo.