Operating Systems Class - Assignment 3

João Ferreira & Pedro Cristina Marques
Department of Informatics Engineering
University of Coimbra
jpbat@student.dei.uc.pt | pgcm@student.dei.uc.pt
2009113274 | 2007184032

December 2011

Introdução

Neste trabalho foi-nos proposto revisitar um conceito abordado no início do semestre, uma impressora virtual. Consistia em receber e montar ficheiros, definidos como `jobs` e, uma vez todo montado, imprimi-lo usando para isso uma abordagem `multithreaded`.

Implementação

São definidas sete constantes, sendo elas:

`MAX_TEXT`
: Define qual o número máximo de bytes do texto de cada segmento. Serve para garantir que as escritas são atómicas. Como o tamanho máximo que um `read` faz é 4096 bytes, então precisou de garantir-se que o tamanho máximo de qualquer segmento enviado nunca ultrapassaria esse valor. Assim, sendo o tamanho base da estrutura 24 bytes, o tamanho máximo to texto será 4060 bytes. No entanto por uma questão de maior facilidade de debug, está por defeito a 20.

`MAX_PAGES`
: Define qual o número máximo de páginas que cada `job`. Assim terá entre 1 e `MAX_PAGES` páginas. Por defeito está a 10.

`N_WORKERS`
: Define o número de `working threads` isto é, o número de `threads` que vão estar a processar os segmentos recebidos.

`N_SENDERS`
: Define o número de `sender threads` isto é, o número de `threads` que vão estar a enviar segmentos para as impressoras.

`N_PRINTERS`
: Define o número de `PRINTERS` isto é, o número de impressoras que vão estar a receber segmentos. O nome das impressoras é: "PRINTERx", em que x é o id correspondente. Por exemplo a impressora com o id 2, tem como nome "PRINTER2".

`DEBUG`
: Define se o programa está a correr, ou não, em `DEBUG MODE`. Caso esta linha se encontre comentada, apenas são efectuados os prints dos `jobs`

já processados. Caso contrário imprime uma série de linhas iniciadas por `[DEBUG]`.

`SLEEP_TIME`
: Define, em segundos, o tempo que cada `sender thread` dorme entre cada escrita nas impressoras, de modo a que seja mais perceptível quais os jobs que estão a ser impressos. Por defeito este valor encontra-se a 3.

Main Thread

Começa por chamar a rotina `init()` que, por sua vez, cria as impressoras, todos os `mutexes`, as `variáveis de condição` e as `queues` de cada `worker thread`. Além disto, arma a protecção para o sinal `SIGINT` que altera a variável `toShutdown` para `true`. Por fim inicializa o `mutex display` que garante o acesso em exclusão mutua ao `stdout` e o `mutex pikachu` que garante o acesso em exclusão mutua à variável `toShutdown`.

Depois disto cria `N_WORKERS threads` que vão tratar dos segmentos recebidos pelo `select()`. Cria ainda `N_SENDERS threads` que vão enviar segmentos de `jobs` através dos vários `named pipes` (printers).

Feito tudo isto começa a correr a função `select()` para receber todos os pedidos enviados e dar trabalho às `worker threads`. Quando o `toShutdown` está a `true`, termina a função `select()` e começa a tratar do `clean shutdown`.

Aqui faz uso da rotina `pthread_join()` para esperar pela morte das `worker threads` e das `sender threads`. Fecha os `file descriptors` das `printers` e remove-as.

Working Threads

Cada `worker thread` inicializa a sua `queue` de trabalhos através da rotina criada `queueReceived_init()`. Em seguida, faz lock ao seu `mutex`, e verifica (em exclusão mútua) se a sua `queue` se encontra vazia, verificando-se esta condição, através da rotina `pthread_cond_wait`, ele fica bloqueado à espera que a `main thread` faça `signal` à variável de condição podendo assim

começar a trabalhar. Verificando-se a existência de `jobs` na `queue` de trabalhos da `thread`, esta faz `dequeue` e adiciona esse mesmo `job` à sua `queue` de `jobs` que estão a ser construidos.

Depois disto, percorrendo toda a sua `queue` de `jobs` verifica quais deles já estão terminados, fazendo `dequeue` desses e imprimindo-os.

`Sender Threads`

Cada `sender thread` começa por aceder em exclusão mútua à variável `toShutdown`, para verificar, se é ou não para continuar a sua execução.

Em seguida, calcula de modo aleatório tanto o número de páginas como o tamanho do texto a enviar em cada uma das páginas, preenchendo-as com texto, também ele aleatório.

Para finalizar envia a informação através de um `named pipe`, fazendo depois um `sleep` de `SLEEP_TIME`.

Conclusão

Podemos assim concluir que as `threads` são uma ferramenta bastante útil se quisermos tirar um melhor partido do nosso computador.

Ainda assim verifica-mos que devem ser usadas com bastante precaução uma vez que, por defeito, a memória é partilhada entre elas podendo levar a erros bastante difíceis de detectar.