Sistemas Operativos 2012/2013 **MIEEC**
Prova de Programação: *Threads* e mecanismos de sincronização POSIX **24 de Abril de 2013**

Duração: 120 + 30 minutos
Consulta: documentação instalada no computador

---

1 Introdução

Esta prova consiste no desenvolvimento dum simulador do jogo ”hungry birds.” (Sim, sabemos que o jogo de que todos falam se chama ”angry birds”, mas este é ainda mais viciante.) A prova está estruturada em 5 alíneas, devendo desenvolver um programa executável em cada uma delas.

Por favor, não se apresse. O tempo previsto deverá ser mais do que suficiente: a nossa solução incremental não tem mais de 80 linhas/instruções novas. Procure resolver cada alínea antes de avançar para a seguinte. Se porventura sentir que está encravado, por favor chame-nos, para o/a aconselharmos sobre a melhor forma de prosseguir.

2 Preparação

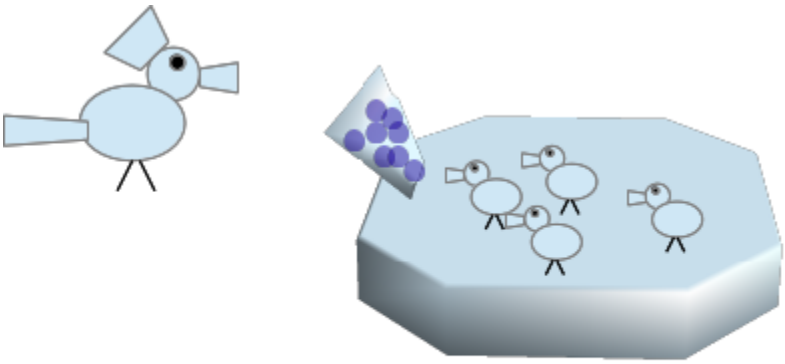
Antes de mais, faça a descompressão do arquivo fornecido, p.ex. usando o comando de linha `unzip pack.zip` num terminal. Esta ação deverá criar um diretório/pasta de nome **pack**, o qual contém este enunciado, o ficheiro com a lista de funções `proto.h` e 3 diretórios:

src o qual contém o código fornecido num único diretório. A resolução de cada alínea poderá ser feita a partir do ficheiro respetivo, `pp[a-e].c`, o qual inclui já código suporte e o esqueleto das soluções a desenvolver.

output o qual contém 1 ficheiro por alínea com a saída gerada pela nossa solução. A saída gerada pela sua solução poderá ser ligeiramente diferente, mas deverá obedecer ao especificado abaixo neste enunciado.

web.fe.up.pt o qual contém uma imagem de parte da documentação disponível na página da disciplina. Pode consultá-la apontando o seu *browser* para o ficheiro:
`pack/web.fe.up.pt/~pfs/aulas/so2013/index.html`
(Deverá preceder esta *string* com o *path name* do diretório onde descomprimiu o arquivo.)

3 O problema dos ”hungry birds”



Um pássaro trata do ninho onde estão B passarinhos esfomeados. Os passarinhos comem de um recipiente em forma de teta que só fornece um bocado de comida de cada vez. Os passarinhos estão sempre esfomeados, pelo que estão sempre a tentar comer um pedaço de comida; quando um o consegue fazer, afasta-se para digerir o bocado (permitindo que outros passarinhos também comam) e torna a tentar ir à teta. Quando um passarinho nota que a teta está vazia, começa a piar para acordar o pássaro e volta a tentar comer. O pássaro, acordado pelos pios, vai buscar comida e reabastece a teta com F bocados enquanto todos os passarinhos aguardam sem comer; depois, avisa os passarinhos de que já há comida na teta e volta a dormir. Este ciclo de actividade continua por R reabastecimentos (e.g. 1 000 000). Nessa altura, o pássaro avisa os passarinhos que chegou a hora de tratarem deles próprios e abandona o ninho. Os passarinhos, já crescidos, vão também embora tratar da vida deles algures.

Represente os passarinhos por B *threads*, `baby()`, o pássaro por um *thread*, `parent()`, e desenvolva código que simule a actividade apresentada. (**Nota:** um outro *thread*, `checker()`, cujo código será fornecido, tratará de verificar a correção da simulação.)

4 Classificação e estrutura da prova

A prova pretende avaliar as capacidades de programação (em C, *threads* e mecanismos POSIX de sincronização); será dividida em várias alíneas (sub-programas) que focarão um dado tipo de técnica de programação; por isso, as primeiras alíneas não constituirão qualquer ”solução” para o problema, mas podem ser encaradas como passos intermédios no processo de produção duma solução, que deverá ser conseguida nas últimas alíneas. A classificação a obter ao longo das alíneas será cumulativa, crescendo com a dificuldade total da programação. Para cada alínea terá que submeter um ficheiro com o código fonte correspondente. A seguir apresenta-se as alíneas da prova com a indicação do objectivo e o valor da classificação máxima que pode ser atingida em cada uma.

a) (20%) `ppa.c` - criar *threads* sem passagem de argumentos e esperar pela sua terminação, sem se receber qualquer dado

b) (40%) `ppb.c` - passar argumentos a *threads* e receber os seus valores de terminação

c) (60%) `ppc.c` - simular o problema dos ”hungry birds”, ignorando condições de competição

d) (80%) `ppd.c` - simular o problema dos "hungry birds", resolvendo as condições de competição com mutexes (e com espera activa)

e) (100%) `ppe.c` - simular o problema dos "hungry birds", resolvendo as condições de competição com mutexes e variáveis de condição (**sem** espera activa). Se preferir pode usar semáforos anónimos POSIX.

Para cada alínea é fornecido um esqueleto do programa respectivo que deverá ser completado com respeito pelas informações locais específicas: variáveis e linhas de impressão de mensagens informativas (e.g. com `printf()`). O não respeito por tais regras e variáveis pode impedir o avaliador de fazer o seu trabalho com alguma automatização e assim considerar errado um programa potencialmente correcto noutro contexto; por exemplo, um programa submetido sem uma dada linha de código de "`printf()`" exigida pode ser considerado incorrecto, porque a mensagem correspondente não foi impressa fazendo assim supor que a computação esperada não era realizada!

A compilação e geração do executável de cada alínea pode ser feita mediante a utilização do "`makefile`" fornecido: na *shell*, escrever "`make pp?",` onde `'?'` é a letra dessa alínea. O teste de executável deve ser feito invocando-o com os parâmetros **nº passarinhos**, B, **nº bocados de comida**, F, **nº reabastecimentos**, R, por esta ordem.

Assim, por exemplo, para desenvolver e testar o programa de resposta à alínea **b**, deve-se:

1. editar e completar "`ppb.c`"
2. invocar "`make ppb`"
3. executar "`./ppb 3 5 10000"` (3 passarinhos, teta com capacidade para 5 bocados de comida, pássaro fará 10000 reabastecimentos.)

5 Pormenores da implementação

As variáveis de programa e as regras de cada alínea têm um fundo comum, que corresponde a variáveis e regras gerais, conducentes à obtenção da "solução real" para o problema dos "hungry birds". Apresenta-se a seguir as principais variáveis e declarações gerais utilizadas nos esqueletos fornecidos e que devem ser respeitados.

Dica: Não se preocupe se numa primeira leitura não compreender o uso de todas as variáveis apresentadas. À medida que for respondendo às diferentes alíneas esse uso tornar-se-á mais claro. Por exemplo, para resolver a alínea **a**, precisa conhecer apenas as funções a executar por cada um dos *threads*. Já na alínea **b**, precisará de entender as estruturas de dados usadas para passar informação aos diferentes *threads*, mas **não** precisa usar (e por isso entender) as variáveis globais `finish` ou `foodbits`. Estas variáveis só são relevantes na alínea **c** e seguintes, quando tem que simular o problema dos "hungry birds".

Variáveis e declarações globais

`MAXBABIES (int)` nº máximo de passarinhos a simular

`finish (int)` iniciado a 0, é colocado a 1 pelo pássaro quando já completou todos os reabastecimentos

`foodbits (int)` nº corrente de bocados de comida na teta

`checkerarg_t` `struct` para passar informação para o *thread* `checker()`

`parentarg_t` `struct` para passar informação para o *thread* `parent()`

`babyarg_t` `struct` para passar informação para os *thread* `baby()`

`parent()` função principal a executar pelo *thread* do pássaro

`baby()` função principal a executar por cada *thread* passarinho

`checker()` função principal a executar pelo *thread* de verificação, cujo código já está feito e compilado (`checker.o`). Contudo, o *thread* tem de ser criado, antes dos outros *threads*, mas não precisa de ser esperado pois é "*detached*".

Nota Importante quando necessitar de usar variáveis de sincronização (mutexes, variáveis de condição, semáforos), defina-as na zona global!

Variáveis locais ao `main()` algumas das quais terão de ser passadas a alguns *threads*:

`B (int)` nº de passarinhos a simular, dado na linha comando

`F (int)` nº de bocados de comida a reabastecer à teta, dado na linha comando e passado aos *threads* `parent()` e `checker()`

`R (long)` nº de reabastecimentos da teta, dado na linha de comando e passado aos *threads* `parent()` e `checker()`

`working (int)` iniciado a 0, é colocado a 1 quando o pássaro trabalha (e os passarinhos esperam, sem comer) e a 0 quando dorme

`eating (int)` iniciado a 0, é o nº de passarinhos que estão a comer em cada instante (deverá ser no máximo 1, pois a teta só permite aceder a um bocado de comida de cada vez)

6 Submissão da Prova

Para terminar a prova deverá:

1. Executar `make clean` no diretório `pack/src`, para reduzir o tamanho do ficheiro a submeter;
2. Comprimir num ficheiro `.zip` único o diretório `pack/src`;
3. Submeter o arquivo resultante através da interface *web* do sistema de gestão de exames (SIGEX), `http://sigex.fe.up.pt/`, para o que será necessário fazer a autenticação com o nome e a senha do FEUPSIG e utilizar o código público desta prova, que depende da sala onde a realizou:

B208 `GOW283` **B213** `FDE103`

Estes passos assumem que respondeu a cada uma das alíneas no ficheiro respetivo do diretório `pack/src`, de acordo com instruções que constam deste enunciado.