Simulação de acesso a recurso partilhado

Metas de aprendizagem

Completando com sucesso o trabalho, os alunos demonstram conhecer e saber utilizar a interface programática de UNIX para:

- criar programas multi*thread*;
- promover a intercomunicação entre processos através de canais com nome (*named pipes*);
- evitar conflitos entre entidades concorrentes, por via de mecanismos de sincronização.

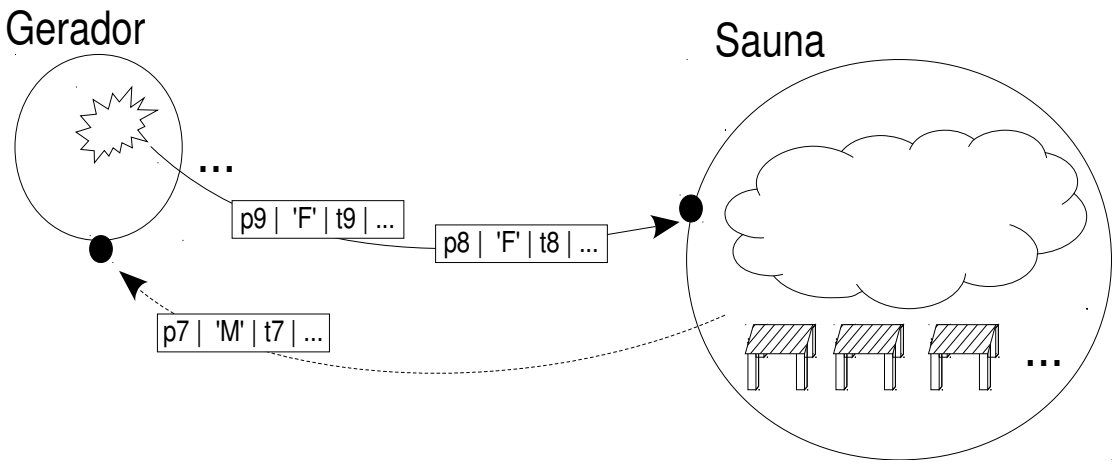
Descrição geral

Pretende-se desenvolver um programa de simulação de acesso a um recurso partilhado. O recurso é uma Sauna unisexo e o programa será uma simulação informática da sua utilização.

O acesso à sauna é controlado um processo que atende pedidos identificados pelo género do cliente e pelo tempo estimado de duração da ocupação. A sauna tem um certo número de lugares disponíveis, que só podem ser ocupados por clientes do mesmo género. Após a admissão de um utilizador à sauna, o programa controla-lhe o tempo de utilização, libertando depois o lugar para outro utilizador.

A simulação de pedidos é efectuada por um programa gerador que sucessivamente emite pedidos caracterizados por um género e por um tempo de ocupação, ambos gerados aleatoriamente. No caso em que um pedido de acesso não pode ser satisfeito de imediato – por exemplo quando o pedido for masculino, `M`, e todos os lugares da sauna estiverem ocupados por utilizadores femininos, `F` –, o pedido é rejeitado e "devolvido" ao gerador que o recolocará na fila de pedidos de acesso.

Todos os pedidos e acessos à sauna são registados em ficheiros, por forma a se poder posteriormente avaliar da correcta execução da simulação.



Requisitos (operacionais e arquitecturais)

Como requisito base:

- os programas produzidos devem executar harmoniosamente, evitando conflitos entre entidades concorrentes no acesso a elementos partilhados.

O programa gerador:

- arranca com o comando
  - `gerador <n. pedidos> <max. utilização>`
    em que
    - `<n. pedidos>` é o número total de pedidos gerados ao longo da execução do programa; atingido esse número, o programa termina graciosamente (ver à frente)
    - `<max. utilização>` é o tempo máximo de duração de uma utilização da sauna, em milisegundos
- contacta o programa que gere a sauna através de um canal com nome (*named pipe*, FIFO), `/tmp/entrada`, em que cada mensagem-pedido tem (pelo menos) os seguintes elementos:
  - *p*, número de série do pedido
  - *g*, género do utilizador que pede acesso ('`F`' ou '`M`')
  - *t*, duração da utilização pedida
- recebe os pedidos rejeitados pela sauna através do canal com nome, `/tmp/rejeitados`, e, em certas condições, recondu-los à sauna (ver à frente)
- é um programa *multithread*, em que
  - um *thread* efectua a geração aleatória de pedidos (tanto relativamente ao género como à duração da utilização, neste caso sujeita ao máximo indicado na linha de comando) e os apresenta à sauna;
  - outro *thread* escuta os pedidos rejeitados e os recoloca na fila de pedidos, mas só caso o número de rejeições de um dado pedido não exceder 3; se for igual a 3, descarta o pedido!

- Durante toda a operação, o programa gerador emite mensagens de registo, para um ficheiro com o nome `/tmp/ger.pid` (em que `pid` é o identificador do processo), que documentam todo o desenrolar da actividade; no final e antes de terminar, emite para a saída padrão uma última informação estatística que indica o nº de pedidos gerados (total e por género), o nº de rejeições recebidas (total e por género) e o nº de rejeições descartadas (total e por género).
  As mensagens de registo terão o formato, por linha:
  - *inst – pid – p: g – dur – tip*
    em que
    - *inst* é o instante de tempo em que a mensagem foi emitida, medido em milisegundos e com 2 casas decimais, e tendo como referência o instante em que o programa começou a executar
    - *pid* é o identificador do processo
    - *p* é o nº sequencial do pedido
    - *g* é a letra que corresponde ao género do utilizador ('`F`' ou '`M`')
    - *dur* é a duração, em milisegundos, pedida para a utilização de um lugar da sauna
    - *tip* é o identificador do tipo de mensagem: "`PEDIDO`", "`REJEITADO`" ou "`DESCARTADO`"

  - cada campo da linha deverá ter uma largura constante, de modo a facilitar a leitura e interpretação do ficheiro.

O programa que controla o acesso à sauna:

- arranca com o comando:
  - `sauna <n. lugares>`
    em que
    - `<n. lugares>` é o número de lugares que podem ser utilizados em simultâneo por utilizadores do mesmo género
- recebe os pedidos de utilização através do canal com nome `/tmp.entrada` e encaminha-os para os lugares vagos desde que sejam do mesmo género dos lugares já ocupados; caso contrário:
  - se o pedido não puder ser aceite porque é de um utilizador de género diferente do dos lugares já ocupados, rejeita-o, devolvendo-o ao gerador através do canal com nome, `/tmp/rejeitados`
  - se o pedido for do mesmo género do dos utilizadores já na sauna, espera a notificação de que um lugar vagou e aceita o pedido;
- é um programa *multithread*, em que:
  - o *thread* principal efectua a recepção e processamento dos pedidos (acesso à sauna ou rejeição); quando já não houver mais pedidos (FIFO de receção ter fechado no lado de escrita), o *thread* aguarda que todos os seus *threads* completem e imprime as estatísticas indicadas à frente antes de terminar ele mesmo
  - outros *threads*, gerados na hora, esperam pela conclusão da utilização de cada pedido da sauna (tempos recebidos nas mensagens-pedido) e, antes de terminarem, notificam o *thread* principal de que há mais um lugar livre.
- Durante toda a operação, o programa `sauna` emite mensagens de registo, para um ficheiro com o nome `/tmp/bal.pid` (em que `pid` é o identificador do processo), que documentam todo o desenrolar da actividade; no final e antes de terminar, emite para a saída padrão uma última informação estatística que indica o nº de pedidos recebidos (total e por género), o nº de rejeições (total e por género) e o nº de pedidos servidos (total e por género).
  As mensagens de registo terão o formato, por linha:
  - *inst – pid – tid – p: g – dur – tip*
    em que
    - *inst* é o instante de tempo em que a mensagem foi emitida, medido em milisegundos e com 2 casas decimais, e tendo como referência o instante em que o programa começou a executar
    - *pid* é o identificador do processo
    - *tid* é o identificador do *thread*
    - *p* é o nº identificador do pedido
    - *g* é a letra que corresponde ao género do utilizador (`F` ou `M`);
    - *dur* é a duração, em milisegundos, pedida para a utilização de um lugar da sauna
    - *tip* é o identificador do tipo de mensagem: "`RECEBIDO`", "`REJEITADO`" ou "`SERVIDO`" ????

No quer estiver omisso:

- pode o projectista/programador definir a estratégia que julgar ser mais apropriada.

Produto final

O trabalho total consiste na produção de um ficheiro compacto a submeter para avaliação via Moodle, que inclui:

- o código-fonte com os programas desenvolvidos
- um *makefile* preparado para facilitar a geração dos executáveis
- um ficheiro de texto, identificando os elementos do grupo de alunos autores e contendo uma explicação sucinta de como foram evitadas no código desenvolvido as situações de competição (*race conditions*) no acesso a elementos partilhados.
- dois ficheiros de texto, cada um contendo toda a informação respeitante à execução de uma instância de `gerador` e de `sauna` (comando de invocação, informação da saída padrão e registos).

O compacto é identificado com um nome do tipo `TxGyy.tar.gz`, onde `x` e `yy` são o número da turma e do grupo, respetivamente.

Avaliação

Será efectuada através de testes simples de execução, em que as mensagens de monitorização produzidas pela simulação nos ficheiros de registo indicados serão analisadas relativamente a exactidão e consistência.