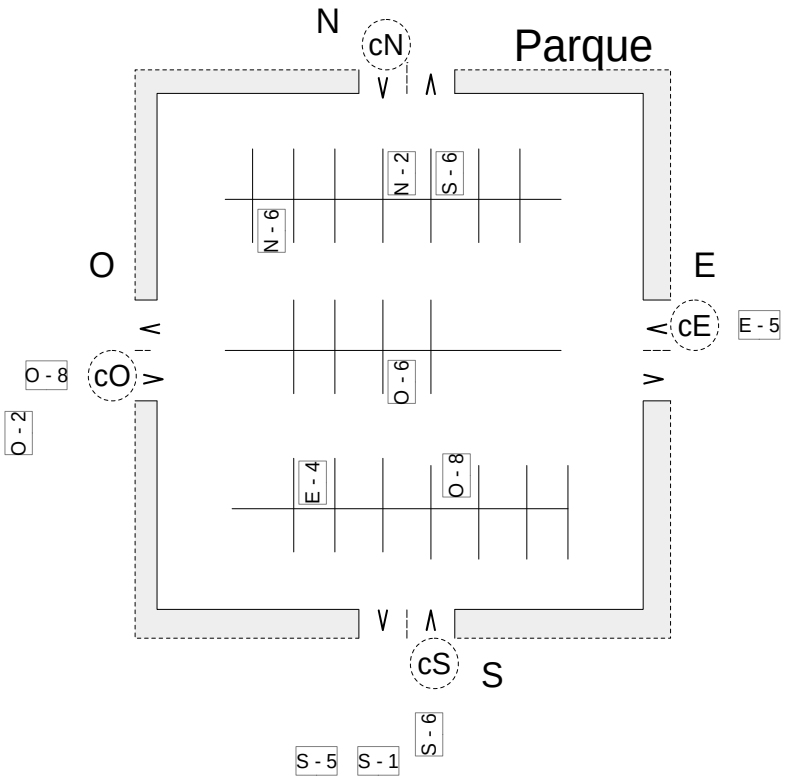
Simulação de Parque de Estacionamento

Objetivos

O objetivo deste trabalho é possibilitar o treino de programação de aplicações Unix/Linux envolvendo processos e *threads*, respetivos meios de intercomunicação e primitivas de sincronização.
Para o efeito, propõe-se o desenvolvimento de uma aplicação que simule o funcionamento de um Parque de Estacionamento.

Descrição geral

Um Parque de Estacionamento com uma dada lotação, tem 4 pontos de acesso (entrada/saída), coincidentes com os pontos cardeais. Em cada ponto está um controlador que atende as viaturas que chegam e encaminha cada uma para um "arrumador"; este, se houver vaga, acompanha a viatura no seu estacionamento durante o tempo desejado; se o Parque estiver cheio, encaminha-a logo para a saída local.
No decurso da simulação, viaturas vão chegando a cada um dos pontos de acesso e recebem o tratamento descrito. Ao fim de um dado intervalo de tempo, o Parque deixa de aceitar viaturas mas mantém-se aberto para saídas enquanto houver viaturas estacionadas; quando todas tiverem saído, encerra completamente.



Arquitetura geral da aplicação

A simulação da operação do Parque decorre mediante o arranque de dois tipos de programas, um ligado ao Parque propriamente dito (**parque**) e outro à geração de viaturas para estacionar (**gerador**). Ambos os programas são *multithread*, recebem parâmetros de configuração pelas respetivas linhas de comando e mantêm-se em execução durante períodos de tempo especificados como parte dos parâmetros de configuração.

A gestão do **Parque** é efetuada por 2 níveis de *thread*s: num nível, dá-se o atendimento das chegadas pelos 4 *thread*s "controladores", cada um associado a um dos pontos de acesso (entrada/saída); a outro nível, dá-se o atendimento "personalizado" de cada viatura, mediante a criação "na hora" de um *thread* "arrumador" que a acompanha durante a sua permanência no Parque: ou a rejeita logo de início ou a deixa estacionar durante o tempo solicitado. Cada evento relevante, e.g. pedido de acesso de uma viatura ou saída de outra, serão anotados num ficheiro de registos, e sempre associados ao instante de ocorrência (em *clock ticks*).

O programa **Gerador**, de forma pseudo-aleatória, "cria" viaturas e associa um novo *thread* a cada uma. Cada *thread* associado a uma viatura (*thread* "viatura") acompanha-la-á no seu "ciclo de vida", desde que é "criada", passando pelo acesso e eventual estacionamento no Parque, até que sai do Parque se nele tiver estacionado e "desaparece".

A **comunicação** entre os processos Parque e Gerador é efetuada por intermédio de *pipes* com nome (FIFOs). Basicamente, cada *thread* "viatura" coloca no acesso estipulado na geração da viatura um pedido de estacionamento, que será recebido pelo *thread* "controlador" desse acesso e respondido (positiva ou negativamente) pelo respetivo *thread* "arrumador" do Parque. O conjunto dos *threads* "arrumador" do Parque (um por viatura que chega ao Parque), coordenam-se na gestão do número de lugares disponíveis por meio de primitivas de sincronização apresentadas nas aulas (mutexes, variáveis de condição, semáforos). Sempre que possível, todas as atividades ligadas à simulação deverão decorrer sem "esperas ativas" (*busy waiting*).

Arquitetura específica dos programas

Parque:

O programa de simulação do funcionamento do Parque é invocado na linha comando por:

```
parque <N_LUGARES> <T_ABERTURA>
```

onde:

- N_LUGARES é a lotação do Parque;
- T_ABERTURA é o período de tempo, em segundos, em que o Parque está aberto; quando vence, não são aceites mais viaturas; quando todas as viaturas tiverem saído, o Parque encerra.

parque é um programa *multithread*, em que, para além do *thread* principal, existem 2 níveis de *threads*: um, corresponde a 4 *threads* "controlador", criados logo de início e que se mantêm nas funções de controlo dos 4 pontos de acesso até ao encerramento do Parque; outro nível, corresponde a *threads* "arrumador", cada um criado por um *thread* "controlador", na sequência da chegada de uma viatura ao respetivo ponto de acesso, e que acompanham a viatura no Parque, desde que chega até que sai (tendo ou não ficado estacionada, dependendo da ocupação do Parque). Todos estes *threads* "arrumador", enquanto estão em execução, coordenam-se na gestão do número de lugares disponíveis do Parque.

O *thread* principal:

- inicializa as variáveis globais necessárias, incluindo o temporizador geral que controla o tempo de abertura do Parque;
- cria os 4 *threads* "controlador", um para cada acesso, e aguarda que eles terminem;
- finalmente, efetua e publica estatísticas globais.

Cada *thread* "controlador", associado a um ponto cardeal de acesso, tem a seu cargo as seguintes tarefas:

- criar o seu FIFO próprio (identificado por "fifo?", onde '?' será ou N ou E, ou S ou O);
- receber pedidos de acesso através do seu FIFO; cada pedido inclui os seguintes dados:
  - porta de entrada;
  - tempo de estacionamento (em *clock ticks*);
  - número identificador único da viatura (inteiro);
  - nome do FIFO privado para a comunicação com o *thread* "viatura" do programa Gerador.
- criar um *thread* "arrumador" para cada pedido de entrada recebido e passar-lhe a informação correspondente a esse pedido;
- estar atento a uma condição de terminação (correspondendo à passagem do T_ABERTURA do Parque) e, nessa altura, ler todos os pedidos pendentes no seu FIFO e fechá-lo para que potenciais novos clientes de estacionamento sejam notificados do encerramento do Parque; encaminhar os últimos pedidos a correspondentes *thread* "arrumador".

Cada *thread* "arrumador" (do tipo *detached*), criado na sequência da chegada de uma viatura pedindo estacionamento, tem a seu cargo as seguintes tarefas:

- recolher a informação sobre a viatura de que está encarregue;
- verificar se há lugar para o estacionamento da viatura ou não (nota: esta operação deverá ser executada de forma sincronizada com todos os outros *thread*s do mesmo tipo, não havendo competição por um lugar concreto do Parque!)
  - se não houver vaga (Parque cheio), nega à viatura o acesso ao Parque, mediante a colocação de uma mensagem disso indicativa ("cheio!") no FIFO privado da viatura, que será lida pelo respetivo *thread* "viatura" do programa Gerador;
  - se houver vaga no Parque, reserva-a (evitando condições de competição – *race conditions* – com todos os outros *thread*s do mesmo tipo) e envia, pelo FIFO privado da viatura, uma mensagem esclarecedora do estacionamento ("entrada") ao respetivo *thread* "viatura"; neste caso a viatura 'desaparece'.
- na sequência da admissão de uma viatura ao Parque, ligar um temporizador local, para controlar o tempo de estacionamento da viatura, e quando o prazo terminar, "acompanhar a viatura à saída do Parque", colocando uma mensagem indicativa ("saida") no FIFO privado da viatura, que será lida pelo respetivo *thread* "viatura";
- na sequência da saída de uma viatura do Parque, dar baixa do lugar ocupado, atualizando (de forma sincronizada com todos os outros *thread*s do mesmo tipo) a contagem do número de lugares vagos no Parque.

Todos os eventos deverão ser registados num ficheiro único, "parque.log", com o formato ilustrado ao lado. Trata-se de um ficheiro CSV para melhor ser importado por uma folha de cálculo e, assim, facilitar a criação de gráficos de visualização do andamento da simulação.

Registo do Parque:

```
t(ticks) ; nlug ; id_viat ; observ
     35  ;    1 ;      1   ; estacionamento
     39  ;    2 ;      2   ; estacionamento
     41  ;    3 ;      1   ; saida
...
   2539  ;   68 ;     23   ; cheio
   2590  ;   67 ;      9   ; encerrado
...
```

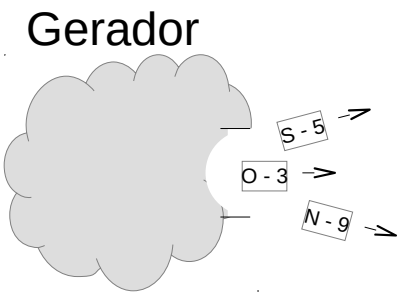
Gerador:

O programa que vai gerar os eventos de "criação" de viaturas é invocado na linha comando por:

gerador <T_GERACAO> <U_RELOGIO>

onde:

- T_GERACAO é o período de tempo, em segundos, em que o Gerador irá funcionar;
- U_RELOGIO é a unidade geral de tempo; o intervalo de tempo mínimo entre quaisquer dois eventos que devem ser medidos (em tiques de relógio). Por exemplo, se for 10 *ticks*, a duração de um estacionamento só poderá ser 10, 20, 30... *ticks*.



gerador é um programa *multithread*, em que o *thread* principal:

- depois de preparar todas as variáveis globais relativas ao funcionamento do processo,
- gera de forma pseudo-aleatória dados que indiquem quando irá ser criada uma nova viatura, para que acesso do Parque se irá dirigir (N, E, S, O) e quanto tempo irá estar estacionada; calcula também um número identificador da viatura (único, portanto);
  - nota: a probabilidade de se selecionar qualquer acesso é idêntica; os tempos de estacionamento devem estar compreendidos entre 1 e 10 múltiplos de U_RELOGIO com idêntica probabilidade; o intervalo entre geração de viaturas deve ser 0, 1 ou 2 múltiplos de U_RELOGIO com probabilidades de 50%, 30% e 20% respetivamente;
- seguidamente cria um novo *thread* "viatura" (no instante determinado de criação) que ficará encarregue do "ciclo de vida" da viatura, e passa-lhe toda a informação pertinente;
- finalmente, aguarda que termine um temporizador próprio, iniciado a T_GERACAO, e termina.

Cada *thread* "viatura" (do tipo "*detached*"):

- recebe os dados relativos à sua viatura do *thread* principal (como argumentos da função do *thread*);
- cria um FIFO privado, de nome único;
- escreve todos os dados relativos à viatura (e que permitirão o acompanhamento do seu ciclo de vida) no correspondente FIFO do Parque (fifoN, fifoE, etc.).
- Seguidamente aguardará, bloqueado no seu FIFO privado, a indicação de que a viatura já saiu do Parque, de que não foi aceite por o Parque estar cheio ou de que o Parque encerrou e não aceita mais viaturas.
- Todos os eventos deverão ser registados num ficheiro único, "gerador.log", com o formato ilustrado ao lado.

Registo do Gerador:

```
t(ticks) ; id_viat ; destin ; t_estacion ; t_vida ; observ
     45  ;    1    ;    N   ;     10     ;    ?   ; entrada
     47  ;    2    ;    S   ;     15     ;    ?   ; entrada
     49  ;    3    ;    N   ;     43     ;    ?   ; cheio!
...
    589  ;    1    ;    N   ;     10     ;   15   ; saída
    591  ;   13    ;    O   ;     22     ;   34   ; saída
...
    989  ;   47    ;    E   ;     30     ;   ?    ;
```

Entrega do trabalho

- Data limite para a entrega do trabalho: **2016/05/22, às 23:55h**.
- A forma de organização dos ficheiros deverá ser semelhante à indicada para o 1º trabalho prático, publicada, na página de "Sistemas Operativos", MIEIC, no Moodle da Universidade do Porto. Deve ser incluída uma *makefile* para a compilação dos programas.
- Deverá ser incluído um exemplo de um ficheiro de "Registos do Gerador" e os correspondentes "Registos de Parque", com indicação das condições em que foi obtido (comandos usados no arranque do parque e do gerador).