EP2 MAC0422 2017/2

Guilherme Costa Vieira
Nº USP: 9790930
Victor Chiaradia Gramuglia Araujo
Nº USP: 9793756

Globals.h:

- Possui definições importantes que são usadas por todos os arquivos diferentes, uint, position e rank.

{position pos; uint laps, broken_lap, pts, tag; double time_elapsed; } rank;

emalloc(size_t size);

A pista:

- A pista: Uma matriz d x LANES (10) global da struct square (um mutex junto comum unsigned int) com funções:
  - initializeTrack();
  - updateTrack();
  - destroyTrack().

```
typedef struct {

        uint pos;

        pthread_mutex_t mutex;

        } square;
```

linked_list.h e sort.h:

Possuem funções feitas para ajudar com a saída do programa:

- printLap(): Imprime a ordem em que os ciclistas terminaram uma volta.
- sort_range_array(): Ordena uma parte delimitada de um vetor.
- Funções de comparação para qsort().
- Funções criar, inserir, remover, destruir, etc de lista ligada.

A simulação:

Discretização do movimento:

Como a taxa de atualização é de 60ms/20ms e cada índice da matriz representa 1 metro, não é sempre que um ciclista irá se mover, assim para contornar esse problema cada ciclista tem um contador “de metro” chamado updatePos.

Quando a operação (updatePos + ((int) (velInRefreshTime(velocity, refresh)*refresh)))%refresh retornar 0 o ciclista irá tentar se mover.

A simulação:

Movimento:

Cada posição da pista possui um mutex.

Ao iniciar uma atualização, as posições vazias estão unlocked e as posições com ciclistas estão locked pelo ciclista que ocupa essa posição.

O ciclista não dá unlock na sua posição enquanto ele não se mover (ou não).

Ao olhar as posições em que o ciclista pode ir, ele tem que dar lock nessas posições. Se houver alguém na posição em que o ciclista pretende ir, significa que o ciclista que ocupava a posição não se moveu e portanto o ciclista não pode ocupar essa posição e logo em seguida dá.

A simulação:

Pontuação:

Sempre que um ciclista completar uma volta, o ciclista irá acessar um vetor global com 4 posições que correspondem quantas voltas ele deve ter completado para ganhar a pontuação associada aquele índice, após aumentar sua pontuação, ele irá aumentar o valor presente na posição acessada do vetor.

Para o ganho dos 20 pontos adicionais, a thread report() encarregada da saída do programa irá ver se o primeiro colocado possui certo número x de voltas a mais que o segundo colocado e se sua pos->x é menor que a dele, se sim o primeiro ganha 20 pontos e x é incrementado.

A simulação:

- Quebra: após um ciclista quebrar ele muda a tag da sua posição ocupada na pista para zero, irá dar unlock no mutex correspondente e irá sair do loop principal da thread para então chamar um thread dummy cuja função é simplesmente chegar nas barreiras de sincronização.
- 90km/h: A escolha de qual ciclista irá percorrer as duas últimas voltas a 90km/h é feita pela main(), antes que qualquer ciclista seja inicializado. Para alertar que a simulação agora será realizada a 20ms uma variável global é mudada entre as duas barreiras.
- Fim para um ciclista: Quando um ciclista acaba a corrida ele irá chamar uma thread dummy para que ela acesse as barreiras de sincronização.

Experimentos

Os experimentos foram realizados em um computador com processador Intel Core i5 7400 (4 cores)

Os dados de tempo consumido e memória utilizada pelo EP2 foram obtidos através do programa “ $ /usr/bin/time -v”.

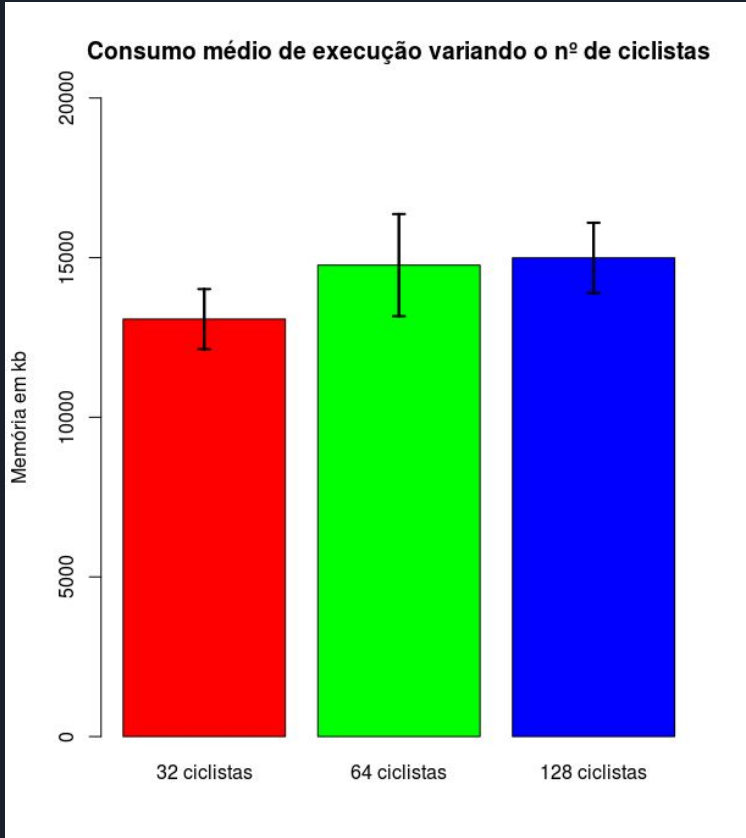
Constantes fixadas para realizar os experimentos:

Nº de voltas: 40

Nº de ciclistas: 64

Tamanho da pista: 250

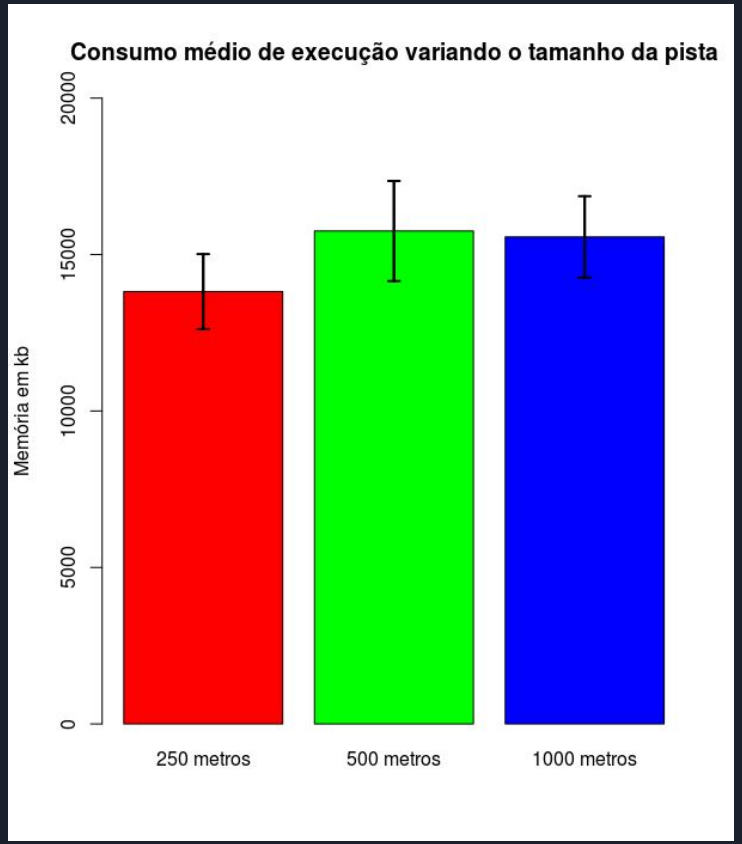
Consumo médio de memória



Confiança de 95%

- Média 32 ciclistas = 13075.7 kb
  - IC = (12135.7kb, 14015.7kb)
- Média 64 ciclistas = 14765.7b
  - IC = (13165.7kb, 16365.7kb)
- Média128 ciclistas = 14993.9kb
  - IC = (13893.9kb, 16093.9kb)

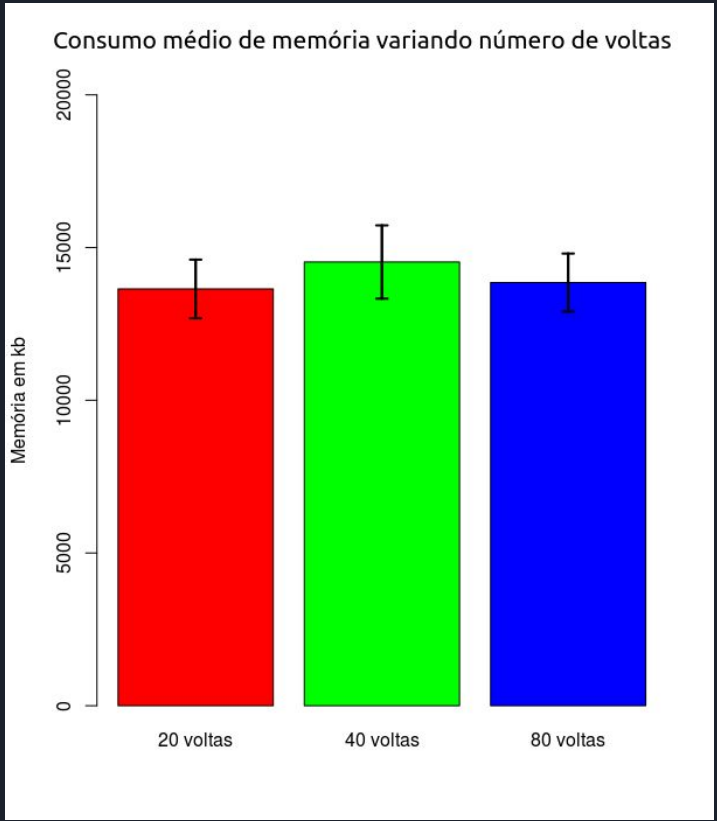
Consumo médio de memória



Confiança de 95%

- Média 250 metros = 13816.8kb
  - IC = (12616.8kb, 15016.8kb)
- Média 500 metros = 15753kb
  - IC = (14153kb, 17353kb)
- Média 1000 metros = 15564.8kb
  - IC = (14264.8kb, 16864.8kb)

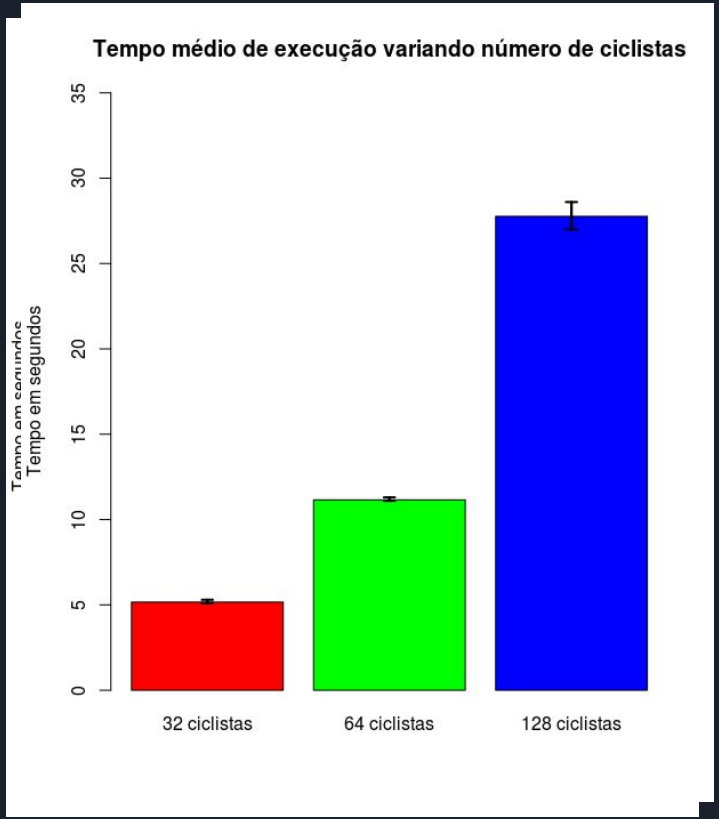
Consumo médio de memória



Confiança de 95%

- Média 20 voltas = 13646.2 kb
  - IC = (12686.2kb, 14606.2kb)
- Média 40 voltas = 14527.5kb
  - IC = (13327.5kb, 15727.5kb)
- Média 80 voltas = 13855.7kb
  - IC = (12905.7kb, 14805.7kb)

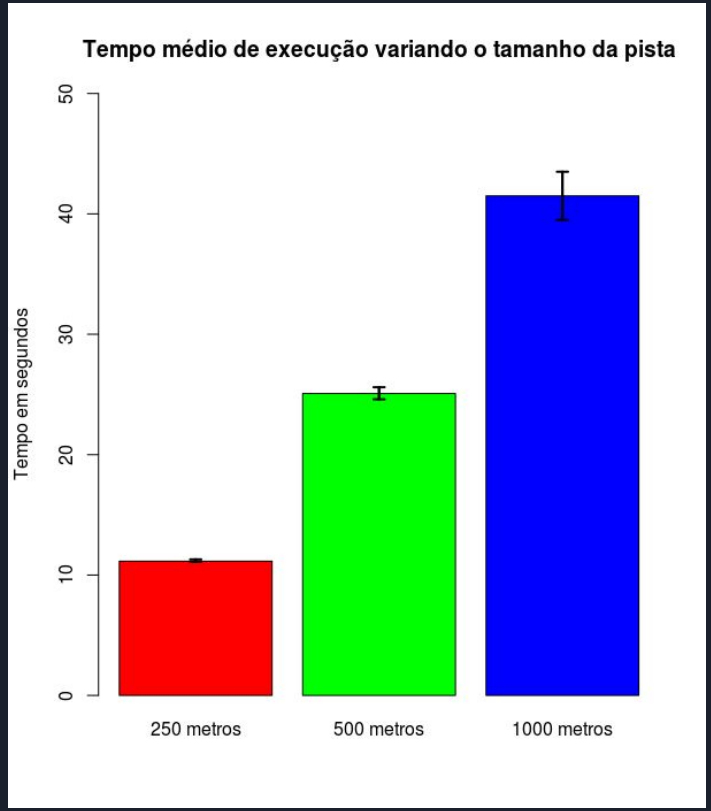
Tempo médio de execução



Confiança de 95%

- Média 32 ciclistas = 5.2s
  - IC = (5.1s, 5.3s)
- Média 64 ciclistas = 11.2s
  - IC = (11.1s, 11.3s)
- Média 128 ciclistas = 27.8s
  - IC = (27s, 28.6s)

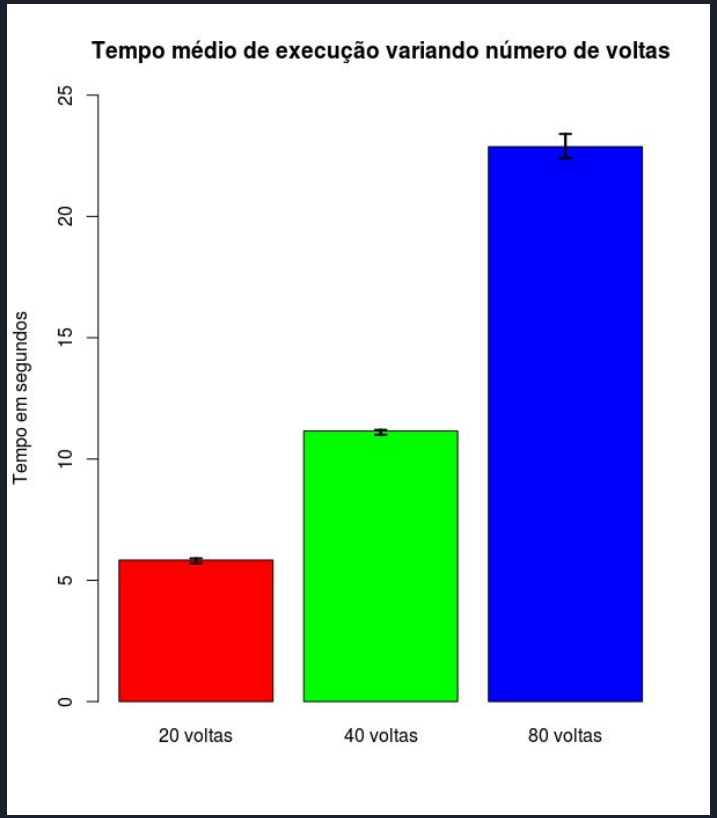
Tempo médio de execução



Confiança de 95%

- Média 250 metros = 11.2s
  - IC = (11.1s, 11.3s)
- Média 500 metros = 25.1
  - IC = (24.6s, 25.6s)
- Média 1000 = 41.5s
  - IC = (39.5s, 43.5s)

Tempo médio de execução



Confiança de 95%

- Média 20 voltas = 5.8s
  - IC = (5.7s, 5.9s)
- Média 40 voltas = 11.1s
  - IC = (11s, 11.2s)
- Média 80 voltas = 22.9
  - IC = (22.4s, 23.4s)