**MAC 438 – Programação Concorrente**
**Prof. Marcel Parolin Jackowski**
**Departamento de Ciência da Computação**
IME/USP – Primeiro Semestre de 2016
**Primeiro Exercício-Programa**
Data de entrega: até 28/03/2016 às 23h55

# Introdução

Este exercício-programa tem o objetivo de tornar natural o uso de algumas *system calls* do Linux, particularmente `fork()` e `wait()`, relacionadas a controle de processos.

O seu programa deverá receber quatro inteiros positivos $m$, $n$, $r$ e $s$ da linha de comando. Deverá, então, iniciar quatro processos (`fork()`), esperar que eles terminem (`wait()`) e encerrar. Os processos a serem criados serão descritos a seguir.

Você deverá imprimir na tela quando um processo iniciou e quando ele terminou, bem como quando o programa principal começou a espera e quando ele foi encerrado. (Rodando o programa várias vezes, você irá notar que essa ordem é imprevisível!)

## $P_1$ – Ordenação

O primeiro processo deverá ordenar $m$ inteiros aleatórios utilizando *heap sort*. Você poderá utilizar algum livro-texto para realizar a implementação, mas deverá deixar a referência utilizada explícita em comentários e/ou num arquivo `LEIAME`. Os inteiros deverão estar entre 0 e 99 e ser gerados por meio das funções `srand()` e `rand()` num array *local* (processo filho). O processo deverá mostrar na tela a lista dos inteiros gerados, ordená-los e, então, mostrar a sequência resultante da ordenação.

## $P_2$ – Números de Fibonacci

O segundo processo deverá computar o $n$-ésimo número de Fibonacci $f_n$. Embora existam várias formulações matemáticas para isso, a implementação deverá empregar a forma recursiva: $f_n = f_{n-1} + f_{n-2}$, para $n > 2$, com $f_1 = f_2 = 1$ (Sim, queremos consumir tempo!). O processo deverá mostrar na tela para qual $n$ está calculando o número de Fibonacci e, então, o seu valor.

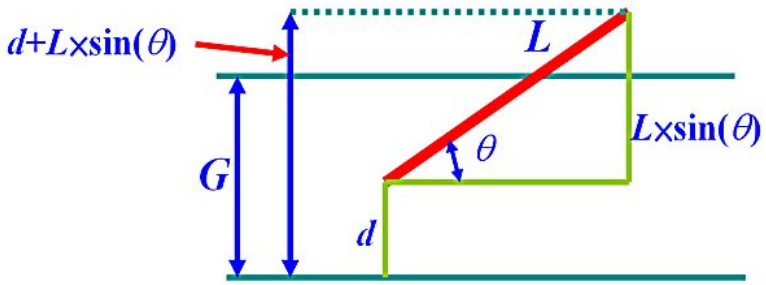
## $P_3$ – Agulha de Buffon

O terceiro processo deverá resolver o problema da agulha de Buffon em $r$ lançamentos. O problema foi formulado pelo matemático francês George-Louis Leclerc, Conde de Buffon, em 1733. Mas este encontrou a solução apenas em 1777.

Suponha que o chão possa ser dividido por um número infinito de retas paralelas, com intervalo delimitado pela constante $G$. Se jogarmos uma agulha de comprimento $L$ no chão aleatoriamente, qual é a probabilidade de a agulha cruzar uma das retas? A resposta é $\frac{2}{\pi} \cdot \frac{L}{G}$.

O seu programa deverá simular esse processo a partir de um laço de $r$ iterações. Se a agulha cruzar uma reta $t$ vezes, então $t/r$ é uma aproximação para a probabilidade exata. De fato, para $r$ "muito grande", o valor simulado deve ser bastante próximo do valor exato. Por simplicidade, considere $L = G = 1$. Para tanto, você precisará gerar dois números aleatórios por iteração, $d \in [0, G)$, que representa a distância de uma das pontas da agulha para a reta inferior, e $\theta \in [0, 2\pi)$ que representa o ângulo da agulha com uma das retas. Se $d + L\sin(\theta)$ for menor que 0 ou maior que $G$, a agulha cruza uma reta.

O processo deverá mostrar na tela quantas iterações irá executar e, por fim, a probabilidade estimada.




### $P_4$ – Integração por simulação

Você aprendeu em Cálculo Diferencial e Integral que a $\int_0^1 \sec(x)dx$ corresponde à área entre a curva e o eixo $x$, como no gráfico a seguir.

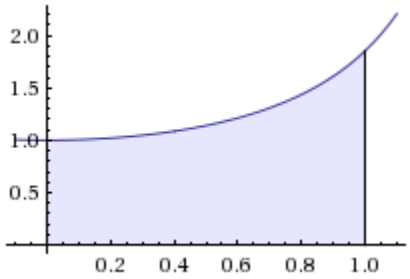



Figura 1: Representação gráfica da área sob $sec(x)$ no intervalo $[0, 1]$

Essa região está completamente compreendida no interior de um retângulo de lados 1 e sec(1), portanto de área sec(1). Sorteando aleatoriamente $s$ pontos $(x_s, y_s)$, com $x_s \in [0, 1]$ e $y_s \in [0, \sec(1)]$, e identificando quantos $t$ pontos destes estão no interior da região hachurada em azul, obteremos a razão $t/s$, que indica a fração da área do retângulo sob a curva da $\sec(x)$. Portanto, $t/s \cdot \sec(1)$ deve ser próximo do valor da integral inicial, especialmente se $s$ for "grande".

O processo deverá imprimir na tela o número de pontos $s$, o número de pontos $t$ e o valor da área estimada.

## Bônus

Apresentamos três pequenos programas abaixo. É esperado um diagrama que evidencie a relação entre os processos pai e filho. Quantos processos são criados em cada caso? Explique passo a passo como os processos são criados e por quem. (Elabore em detalhes para que a sua resposta tenha valor, e não aparente ser um "chute".). A correta análise de cada um vale 1 ponto adicional, e não serão oferecidos décimos (isto é, cada análise terá 0 ou 1 ponto). (Portanto, um EP *bem feito* pode obter 13 pontos!). Entregue um único arquivo PDF com as suas análises, identificando claramente a qual programa se refere.

### Programa 1

```
int main(int argc, char **argv)
{
   int i, n = 4;
   for(i = 1; i < n; i++)
      if (fork())
```

```
         break;
   printf("Processo %d de pai %d\n", getpid(), getppid());
   sleep(1);
   return 0;
}
```

### Programa 2

```
int main(int argc, char **argv)
{
   int i, n = 4;
   for(i = 0; i < n; i++)
      if (fork() <= 0)
         break;
   printf("Processo %d de pai %d\n", getpid(), getppid());
   sleep(1);
   return 0;
}
```

### Programa 3

```
int main(int argc, char **argv)
{
   int i, n = 3;
   for(i = 0; i < n; i++)
      if (fork() == -1)
         break;
   printf("Processo %d de pai %d\n", getpid(), getppid());
   sleep(1);
   return 0;
}
```

## Observações finais

Todo o EP deve ser escrito em C, sem a utilização de bibliotecas externas. É fortemente recomendada a utilização de interfaces (*header files*, arquivos .h) e a organização do EP em módulos, que separem a definição da estrutura de dados e métodos utilizados. Isso ajudará principalmente *você*.

### Sobre a elaboração:

Este EP pode ser elaborado por equipes de um ou dois alunos, desde que sejam respeitadas as seguintes regras:

- Os alunos devem trabalhar sempre juntos cooperativamente.
- Caso em um grupo exista um aluno com maior facilidade, este deve explicar as decisões tomadas. E o seu par deve participar e se esforçar para entender o desenvolvimento do programa (chamamos isso de *programação em pares*, que é uma excelente prática que vocês devem se esforçar para adotar).
- Mesmo a digitação do EP deve ser feita em grupo, enquanto um digita, o outro fica acompanhando o trabalho.

**Sobre a avaliação:**

- É sua responsabilidade manter o código do seu EP em sigilo, ou seja, apenas você e seu par devem ter acesso ao código.
- **Não serão toleradas cópias!** Exercícios copiados (com ou sem eventuais disfarces) levarão à reprovação da disciplina e o encaminhamento do caso para a Comissão de Graduação.
- Exercícios com erros de sintaxe (ou seja, erros de compilação) receberão nota zero.
- É muito importante que seu programa seja elegante, claro e bem indentado, ou seja, digitado de maneira a ressaltar a estrutura de subordinação dos comandos do programa. A qualidade do seu trabalho sob esse ponto de vista influenciará sua nota!
- As informações impressas pelo seu programa na tela devem aparecer da forma mais clara possível. Este aspecto também será levado em consideração no cálculo da sua nota.
- Uma regra básica é a seguinte: do ponto de vista do monitor responsável pela correção dos trabalhos, quanto mais convenientemente apresentado estiver o seu programa, melhor avaliado será seu trabalho.

**Sobre a entrega:**

- Entregar apenas um arquivo de nome `EP1.zip`, contendo todos os arquivos de seu exercício. Se quiser entregar também um arquivo texto contendo uma explicação sobre o programa e referências utilizadas (livros, internet etc.), dê a ele o nome `LEIAME`. Caso o seu programa tenha vários módulos, não se esqueça de incluir um Makefile.
- No início de cada arquivo, acrescente um cabeçalho bem informativo, como o seguinte:

```
/*****************************************************************/
/**   MAC 438  - Programação Concorrente                        **/
/**   IME-USP  - Primeiro Semestre de 2016                      **/
/**   Prof. Marcel Parolin Jackowski                            **/
/**                                                             **/
/**   Primeiro Exercício-Programa                               **/
/**   Arquivo: EP1.c                                            **/
/**                                                             **/
/**   <nome do(a) aluno(a)>               <número USP>          **/
/**                                                             **/
/**   <data de entrega>                                         **/
/*****************************************************************/
```

  Não é obrigatório que o cabeçalho seja idêntico a esse, apenas que contenha pelo menos as mesmas informações.

- Para a entrega, utilize exclusivamente o Paca. Você pode entregar várias versões de um mesmo EP até o prazo, mas somente a última será armazenada pelo sistema.
- Não serão aceitas submissões por email ou atrasadas. Não deixe para a última hora, pois o sistema pode ficar congestionado e você poderá não conseguir enviar.
- Guarde uma cópia do seu EP pelo menos até o fim do semestre e, novamente, você é responsável por manter o sigilo de seu código-fonte.