DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
SISTEMAS OPERACIONAIS
2₀ SEM/2020
Trabalho em Grupo – Nr 2
ESTUDO SOBRE SUBPROCESSOS COOPERATIVOS

---

1. Objetivo do Trabalho

Estimular a capacidade do aluno de trabalhar em equipe para organizar, projetar e desenvolver soluções para problemas formulados que envolvam o estudo e o conhecimento sobre subprocessos e threads.

2. Escopo do Trabalho

- ✓ Estudar comandos indicados.
- ✓ Conceber e implementar os algoritmos conforme as questões do item 7 **itens c, d, e**.
- ✓ Preparar um relatório contendo uma descrição sobre os objetivos do trabalho, as premissas consideradas e apresentar a saída da execução dos programas.
- ✓ Gravar um vídeo de até 10 minutos onde todos os participantes do grupo apresentam o trabalho.
- ✓ Os trabalhos devem ser feitos exclusivamente em C.
- ✓ Incluir no relatório um extrato da console de execução dos programas.
- ✓ As avaliações sobre o funcionamento dos programas serão feitas em horário marcado.

3. Equipes de Trabalho

As equipes devem ser formadas com, **no máximo**, 3 (**três**) alunos.

4. Prazo de Entrega do Trabalho

O materiais (código, vídeo e relatório) devem ser postados no GDrive, pasta “Trabalhos de SO - 2020-2” (link: https://drive.google.com/drive/u/1/folders/1NEmrbyFnd6KHB0o8Q_qTwfMJrvUdM7e7) até às **23:59** do dia **03/06/2021** com o título “**Trabalho 2 de SO 2020-2 - Grupo X**”, onde X é o número do grupo, conforme descrito no item 8. As apresentações serão realizadas no dia **08/06/2021**, em horário previamente sorteado para cada grupo.

5. Penalidades

Caso o grupo atrase a entrega do resumo seu grau final sofrerá um decréscimo na razão de 0,5 pontos por dia.

6. Avaliação

Serão considerados os seguintes aspectos:

- ✓ Execução correta dos programas durante a avaliação;
- ✓ Apresentação do relatório que descreve o trabalho;
- ✓ Apresentação (até 10 minutos) do trabalho em vídeo;
- ✓ Entrega pontualmente efetuada no dia estipulado;

- ✓ Apresentação do grupo no dia e horário indicado;
- ✓ Conhecimento utilizado no desenvolvimento do trabalho;
- ✓ Qualquer regra que não seja seguida pelo grupo implicará na perda de 1,0 pontos por regra.

7. Temas para Desenvolvimento

Utilize o ambiente Linux para desenvolver seus programas.

a. Estudo/ /de Comandos

- ✓ Estude os comandos: fork(); exec(); execl(); wait() e exit(); getpid(), getppid(). Porém, não é necessário efetuar nenhum comentário a respeito no relatório.
- ✓ Leia o material sobre Comunicação entre Processos ou execute o comando "man" em ambiente Unix ou Linux.

b. Anexo 1 – Exemplos de uso das funções fork() e exec()

- ✓ Execute os exemplos dados no **Anexo 1**, e descreva no relatório o que será executado, apresentando os resultados obtidos.
- ✓ Para esses exemplos não será cobrada a apresentação.

c. Prog1 (anexo) - Uso dos comandos fork() e exec()

- ✓ Este exercício propõe o entendimento do uso das funções fork() e exec().
- ✓ Prepare o código de prog1.c, segundo os requisitos solicitados.
- ✓ Execute o programa e responda no relatório às questões colocadas no código do programa.
- ✓ Mostre no relatório o conteúdo da console durante a execução.

d. Prog2 – Subprocessos Cooperativos

- ✓ Construa um programa que simula a execução de um interpretador de comandos.
- ✓ Prepare o código do algoritmo apresentado em prog2.c segundo os requisitos solicitados.
- ✓ **Versão 1 do programa**: O comando não possui opções e argumentos
- ✓ **Versão 2 do programa**: Incluir a possibilidade de passar parâmetros e argumentos para o comando a ser executado, conforme sintaxe de comandos Unix:

  $> comando *opções argumentos*

  Opções: precedidas por '-' e seguindo uma letra

  Argumentos: nomes de arquivos

Sugestão: Você pode utilizar a função *getopt()*.

e. Prog3 – Utilização de Semáforos

Suponha os Processos A e B listados a seguir:

```
Variáveis compartilhadas
semaphore mutex=1;
semaphore a=2;
semaphore b=0;

Processo A:
while true do down(a);
      down(mutex);
      print(A);
      print(B);
      up(mutex);
      up(b);
end while

Processo B:
while true do
      down(b);
      down(mutex);
      print(C);
      print(D);
      up(mutex);
      up(a);
end while
```

Realize a implementação em linguagem C do pseudo-código acima. A execução concorrente dos processos A e B produz uma sequência infinita de impressão dos caracteres “A” e “B”. Indique a única sequência possível para execução simultânea dos processos, justificando.

8. Grupos

- ✓ Grupo 1: Roberto Moreira, Pedro Moraes, Tomaz Cuber Guimarães;
- ✓ Grupo 2: Daniel la Rubia Rolim, Maria Eduarda Hamdan Lucena, Yuri Medeiros;
- ✓ Grupo 3: João Felipe Rocha, Marcelo Fonseca, Matheus Feitosa;
- ✓ Grupo 4: Pedro Novaes Possato, Felipe Kuhnert, João Pedro Murtinho;
- ✓ Grupo 5: Daniel Bayerl Vieira, Lucas Senos, Lucas Miranda;
- ✓ Grupo 6: Fernando França, Rodrigo Passos, Bruno Gavarra;

Caso haja alguma alteração nos componentes do grupo, favor informar pelo e-mail: valeriab@ntt.ufrj.br

OBS: A ordem dos grupos não indica a ordem de apresentação em 08/06/2021.

.

BOM TRABALHO

Anexo 1 – Exemplos de uso

(1)

```
main()
{
   int ret1, ret2;
   ret1 = fork();
   ret2 = fork();
   printf("Programa em execução.\n");
}
```

(2)

```
main()
{
   int ret;
   ret = fork();
   if (ret == 0)
      execl("/bin/ls","ls",0);
   else
      printf("Processo continua executando.\n");
}
```

(3)

```
main()
{
   int ret;
   ret = fork();
   if (ret == 0) {
      execl("/bin/ls","ls",0);
      printf("Quando este comando será executado ? \n");
      };
   printf("Por que a função printf anterior não foi executada?\n");
}
```

(4)

```
main()
{
   int ret;
   ret = fork();
   if (ret == 0) {
      execl("/bin/ll","ll",0);
      printf("Por que este comando foi executado ? \n");
      }
   else
      printf("Processo continua executando.\n");
}
```

prog1.c

```
#include <stdio.h>
#include <wait.h>
#include <unistd.h>

int main(void)
{
      int status, id, j;
      ********* Insira um comando para pegar o PID do processo corrente e mostre na
                tela da console.

      if (*** insira um comando para criar um subprocesso) {
            ***** Faça com que o processo pai execute este trecho de código
            ***** Mostre na console o PID do processo pai e do processo filho
            ***** Monte uma mensagem e a envie para o processo filho
            ***** Mostre na tela o texto da mensagem enviada
            ***** Aguarde a resposta do processo filho
            ***** Mostre na tela o texto recebido do processo filho
            ***** Aguarde mensagem do filho e mostre o texto recebido
            ***** Aguarde o término do processo filho
            ***** Informe na tela que o filho terminou e que o processo pai também
            vai encerrar
      } else {
            ***** Faça com que o processo filho execute este trecho de código
            ***** Mostre na tela o PID do processo corrente e do processo pai
            ***** Aguarde a mensagem do processo pai e ao receber mostre o texto na
            tela
            ***** Envie uma mensagem resposta ao pai
            ***** Execute o comando "for" abaixo

            for (j = 0; j <= 10000; i++);
            ***** Envie mensagem ao processo pai com o valor final de "j"
            ***** Execute o comando abaixo e responda às perguntas

            execl("/Bin/ls", "ls", NULL);
            ***** O que acontece após este comando?
            ***** O que pode acontecer se o comando "execl" falhar?
      }
      exit(0);
}
```

prog2.c

```
Início
      Lê linha de comando;
      Enquanto não fim faça
            Início
                  Percorre a linha retirando o nome do comando;
                  Executa um fork para criar um novo processo;
                  Se processo filho então
                        Executa execl especificando o nome do comando  como
                        parâmetro;
                  Senão
                        Inicio
                        Executa wait para esperar que a execução do comando termine;
                        Se codigo retorno = zero então
                              Escreva "Executado com sucesso."
                        Senão
                              Escreva "Código de retorno = ", codigo_retorno;
                        Fim

                  Fim se;
                  Lê linha de comando;
            Fim;
Fim;
```