# Universidade Federal do ABC
# MCTA026-13 - Sistemas Operacionais
# 2019.Q1

**Lista de Exercícios 7**

Prof. Emílio Francesquini

28 de março de 2019

Lista de termos cuja definição você **deve** saber:

- Programa *vs.*Processo
- Seção de texto, seção de dados, monte (*heap*), pilha (*stack*)
- Estados do processo, PCB, troca de contexto
- Escalonamento de processos
- Criação e finalização de processos
- Finalização em cascata, processos zumbis e órfãos
- IPC: memória compartilhada e troca de mensagens

## Exercícios

1. Considere o código a seguir:

```
1  #include <sys/types.h>
2  #include <stdio.h>
3  #include <unistd.h>
4
5  int value = 5;
6
7  int main() {
8      pid t pid;
9      pid = fork();
10     if (pid == 0) {
11         value += 15;
```

```
12          return 0;
13      }
14      else if (pid > 0) {
15          wait(NULL);
16          printf("PARENT: value = %d",value);
17          return 0;
18      }
19  }
```

Qual será o valor impresso pelo programa na saída padrão?

2. Considere o seguinte código

```
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main() {
5      fork();
6      fork();
7      fork();
8      return 0;
9  }
```

Quantos processos ao total estarão em execução ao final das linhas 5, 6 e 7?

3. Quando um processo é criado utilizando a syscall `fork()`, quais dos seguintes estados é compartilhado entre o processo pai e o filho?

   (a) Pilha

   (b) Heap

   (c) Segmentos de memória compartilhada

4. Descreva as ações que são tomadas pelo SO quando há uma troca de contexto.

5. O processador Sun UltraSPARC tem conjuntos de registradores múltiplos. Descreva o que ocorre quando há uma troca de contexto se o novo contexto já estiver carregado nos registradores do processador. O que ocorre se o novo contexto estiver em memória e não nos registradores e todos os registradores estiverem em uso?

6. Construa uma árvore de processos do seu computador. No Linux use o comando `pd -ael` para listar os processos atuais.

7. Considere o código abaixo

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>
int main() {
    pid_t pid;

    pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork Failed");
        return 1;
    } else if (pid == 0) {
        execlp("/bin/ls", "ls", NULL);
        printf("AQUI");
    } else {
        wait(NULL);
        printf("Child Complete");
    }
    return 0;
}
```

Descreva em que casos a linha 13 será alcançada e executada.

8. Considere o código abaixo

```c
#include <sys/types.h>
#include <stdio.h>
#include <unistd.h>

int main() {
    pid t pid, pid1;


    pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork Failed");
        return 1;
    } else if (pid == 0) {
        pid1 = getpid();
        printf("child: pid = %d",pid); /* A */
        printf("child: pid1 = %d",pid1); /* B */
    } else {
        pid1 = getpid();
        printf("parent: pid = %d",pid); /* C */
        printf("parent: pid1 = %d",pid1); /* D */
        wait(NULL);
```

```
22      }
23      return 0;
24  }
```

Identifique os valores de `pid` nas linhas A, B, C e D. Assuma que o valor dos PIDs do pai e do filho são, respectivamente, 2600 e 2603.

9. Dê um exemplo onde o uso de pipes anônimos seja mais conveniente do que o uso de pipes nomeados e um exemplo onde pipes nomeados seja mais conveniente do que pipes anônimos.

10. Considere o programa abaixo e diga qual será a saída nas linhas X e Y.

```
1   #include <sys/types.h>
2   #include <stdio.h>
3   #include <unistd.h>
4
5   #define SIZE 5
6   int nums[SIZE] = {0,1,2,3,4};
7
8   int main() {
9       int i;
10      pid_t pid;
11
12      pid = fork();
13      if (pid == 0) {
14          for (i = 0; i < SIZE; i++) {
15              nums[i] *= -i;
16              printf("CHILD: %d ", nums[i]); /* Linha X */
17          }
18      } else if (pid > 0) {
19          wait(NULL);
20          for (i = 0; i < SIZE; i++)
21              printf("PARENT: %d ", nums[i]); /* Linha Y */
22      }
23
24      return 0;
25  }
```

11. Considerando tanto o em nível de sistema quanto em nível do usuário, quais são as vantagens e desvantagens de:

    (a) Comunicação síncrona e assíncrona

    (b) Buffering automático ou explícito

(c) Passagem por referência ou por cópia

(d) Mensagens de tamanho fixo ou variável