# Universidade Federal de Uberlândia (UFU)

Faculdade de Computação (FACOM)

Disciplina: GBC045 - Sistemas Operacionais

Prof. Rivalino Matias Jr.

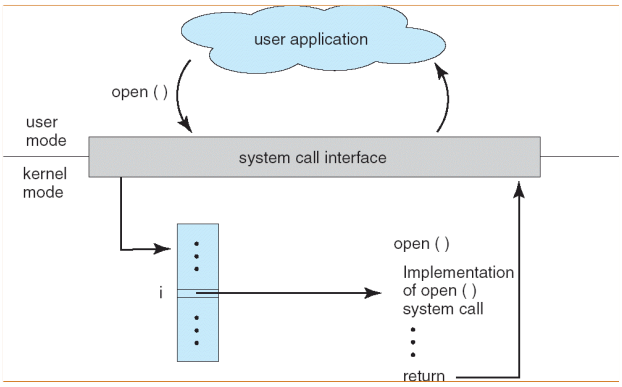
## Exercícios de Fixação (Unidade II)

1) Explique a diferença entre programa e processo através de um exemplo.

2) Cite e explique as partes que compõem um processo.

3) 1) Com base na figura ao lado, explique quando ocorrem as seguintes situações:

a) execução em user level no contexto do processo
b) execução em kernel level no contexto do processo
c) execução em user level no contexto do sistema
d) execução em kernel level no contexto do sistema




4) Um processo cria outro processo usando a chamada de sistemas vfork(2). Em que situações o processo pai encerra sua execução?

5) Desenhe a hierarquia de processos criada a partir da execução do programa abaixo:

```
int main(){
  pid_t pids[2];
  int i;
  for(i=0; i < 2; i++ ){     pids[i] = fork();   }
  getchar();
}
```

6) Execute o programa do exercício (5) em um terminal e em um segundo terminal execute o seguinte programa "*ps aux*" . Com base na saída do progrma *ps*, responda:

a) Qual o PID do processo executando o programa do exercícios 5?
b) Qual o PID do Pai (PPID) do processo executando o programa do exercício 5?
c) Qual o PID de cada filho do processo executando o programa do exercício 5?
d) Quais os estados dos processos envolvidos nos exercícios (a), (b) e (c)?

7) No Linux, após o processo Pai criar um processo Filho, qual processo (Pai ou Filho) será o próximo a ser executado? Faça experimentos de laboratório para justificar sua resposta.

8) Faça um programa que crie um processo Filho e aguarde seu término (dica: use wait() ) para finalizar. O processo Filho por sua vez imprime todos os valores pares de 1 até 1000.

9) Faça um programa que crie dois processos e aguarde seus términos para finalizar. O primeiro processo Filho criado deve imprimir todos os valores pares de 1 até 1000, e o segundo Filho deve imprimir os valores ímpares no mesmo intervalo.

10) Faça um programa que crie 50 processos filhos e aguarde o término de todos para finalizar. Cada processo Filho deve imprimir: *i*) o pid do seu processo Pai e ii) o seu próprio *pid*.

11) Considerando o diagrama de transição de estados genéricos apresentado no slide 10 da Unidade II, descreva os estados e as possíveis transições que podem ocorrer na execução do processo filho criado a partir do programa abaixo.

```
int main(){
  pid_t pid;
  int status;
  pid = fork();
  if       ( pid == 0 )  getchar();
  else if (pid > 0 )     wait(&status);
  exit(1);
}
```

12) Implemente três programas para realizar a computação abaixo.
O **primeiro** deve realizar a computação sem a criação de processos/*threads*.
O **segundo** deve ser baseado em múltiplos processos, onde o processo pai inicializa as matrizes **M** e **N** e cria dois processos filhos, onde cada filho ficará responsável pela computação de $\mathbf{M}^{10}$ e $\mathbf{N}^{10}$. Ao término de cada filho, o processo Pai deve usar os resultados calculados por cada filho para computar **R**.
O **terceiro** deve ser baseado em múltiplas *threads*, onde o processo cria duas *threads*. Uma *thread* ficará responsável pela computação de $\mathbf{M}^{10}$ e a outra pela computação de $\mathbf{N}^{10}$. Ao término de ambas as *threads*, a *thread* principal do processo deve usar os resultados de cada uma das demais *threads* para computar **R**.

$\mathbf{R} = \mathbf{M}^{10} \times \mathbf{N}^{10}$, onde **M** e **N** são matrizes 5000 x 5000 inicializadas com valores inteiros aleatórios.

Execute cada um dos três programas e avalie:

a) O tempo de execução de cada um (dica: use o comando *time*)
b) O consumo dea memória principal por cada programa (dica: use os comandos ps aux ou top ).
c) A dificuldade de programação.