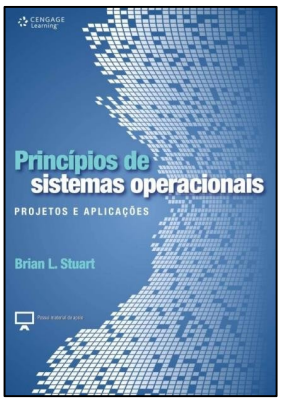
FACULDADE DE COMPUTAÇÃO E INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
SISTEMAS OPERACIONAIS – Aula 04 – 1º SEMESTRE/2020
PROF. LUCIANO SILVA

TEORIA: GERENCIADOR DE PROCESSOS (PARTE III)

Nossos **objetivos** nesta aula são:

- conhecer o mecanismo de criação de processos com chamadas do sistema fork()
- conhecer o conceito de threads e seu mecanismo de criação
- praticar com criação de processos com fork() e threads



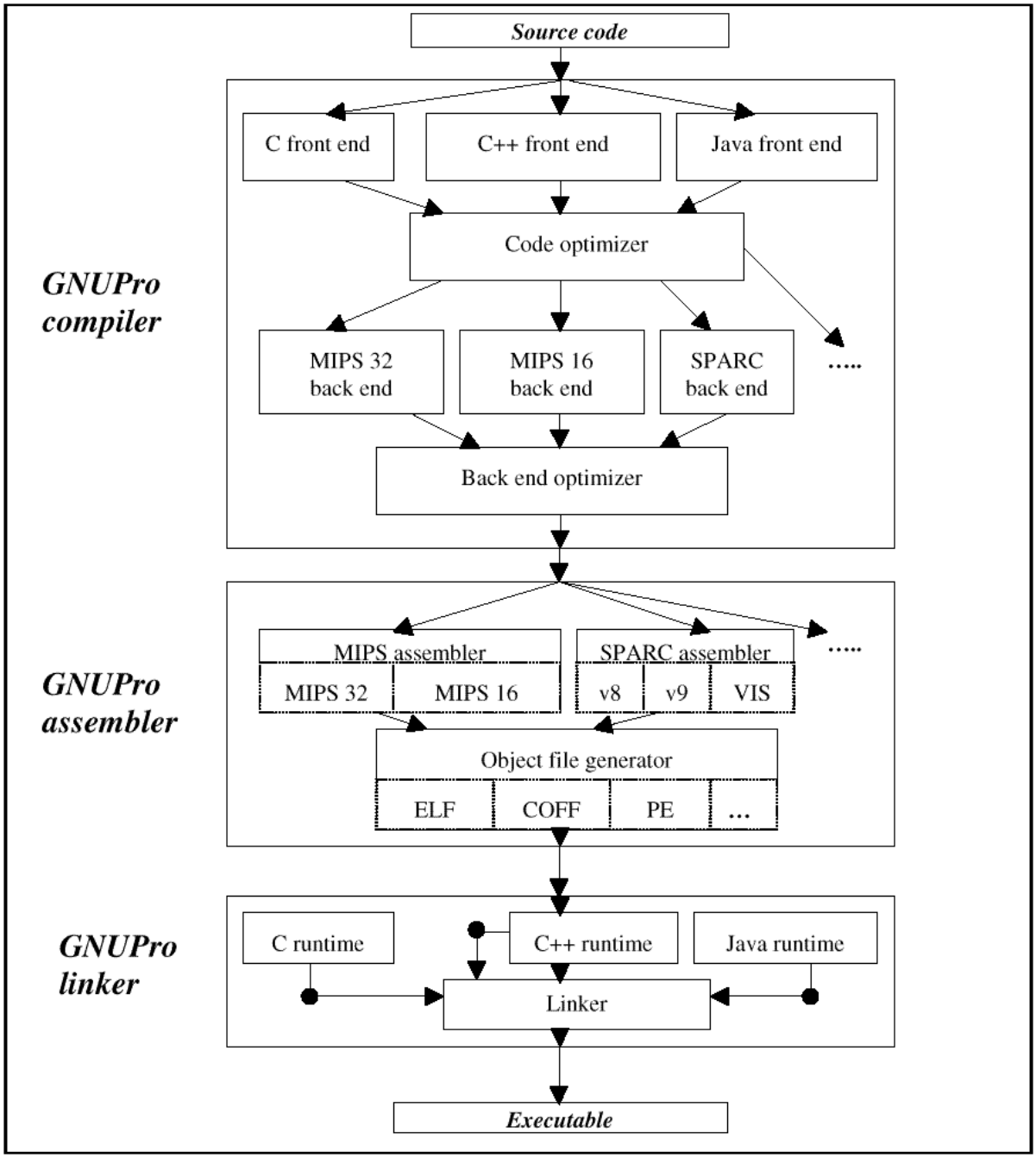
Para esta aula, usamos como referência a **Seção 5.6 (Criação e finalização de processos)** do nosso livro-texto:

STUART, B.L., **Princípios de Sistemas Operacionais: Projetos e Aplicações**. São Paulo: Cengage Learning, 2011.

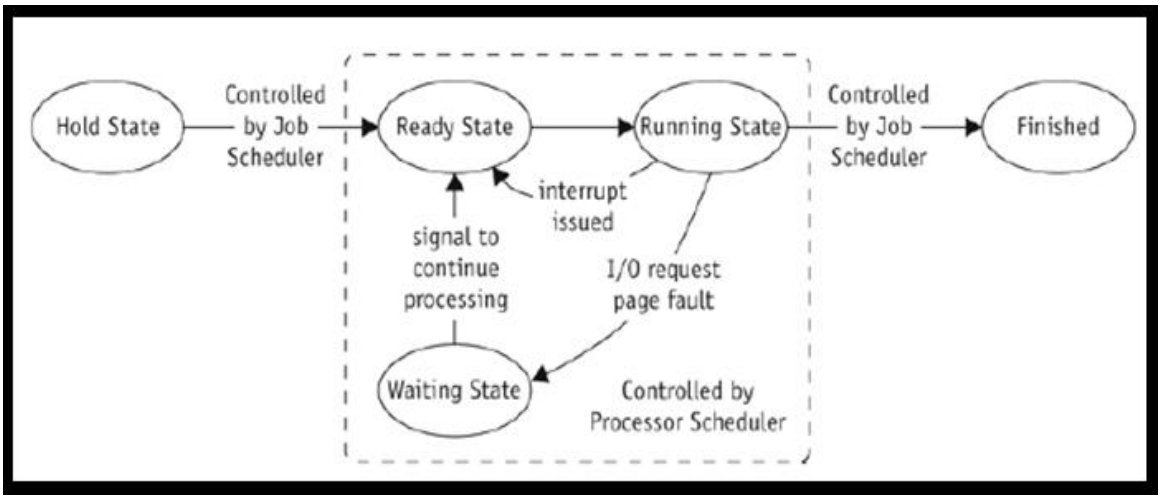
Não deixem de ler esta seção depois desta aula!

PROGRAMAS EXECUTÁVEIS NO FORMATO ELF

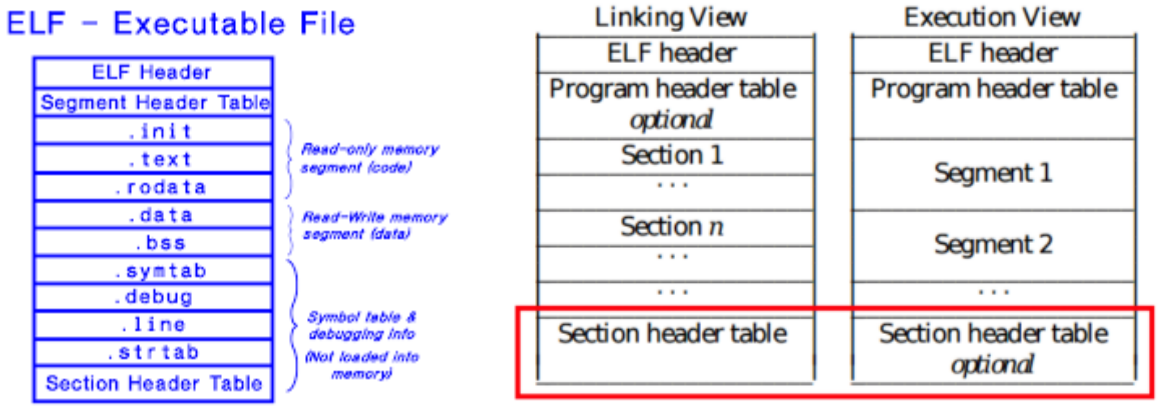
- Sabemos, das aulas anteriores, que um processo é um programa em execução. Antes de virar um executável, um programa .c (por exemplo) para por uma série de fases, até virar um executável (formato ELF, em Linux, Unix e MacOS).



- Quando um programa começa a ser "rodado", ele configura-se como um job e, posteriormente, para processo, quando entra no ciclo de execução.



- Em UNIX, Linux, MacOS, que seguem padrões típicos de UNIX, um formato especial para programas executáveis chamado ELF.

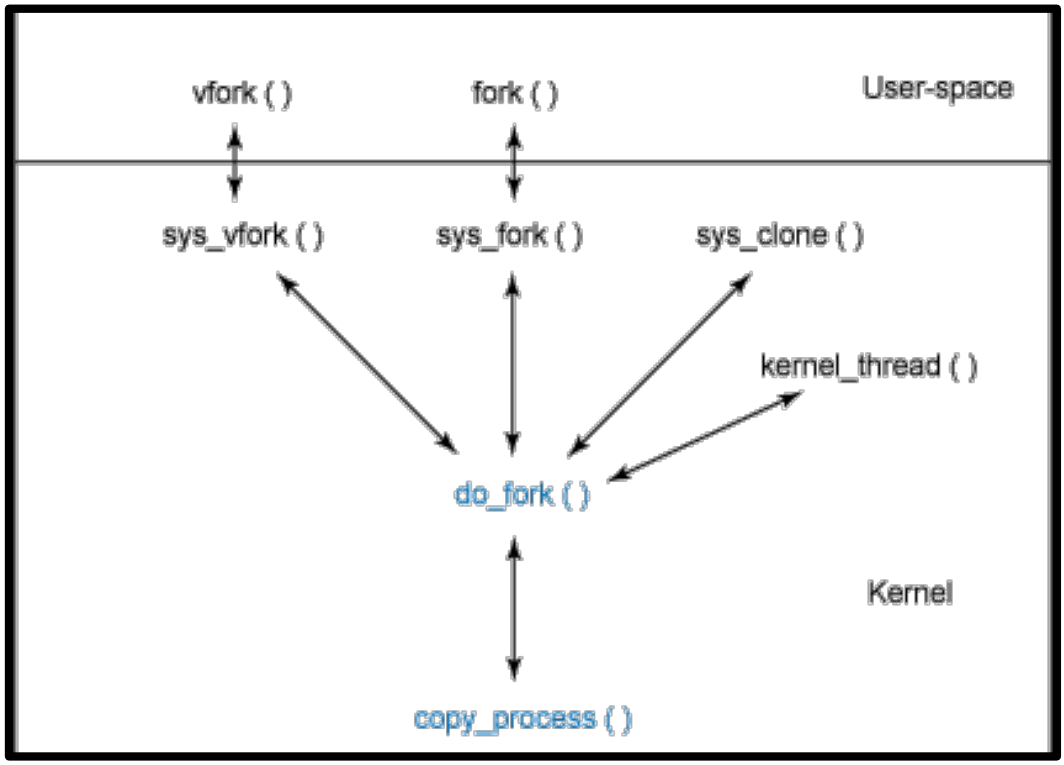


EXERCÍCIO COM DISCUSSÃO EM DUPLAS

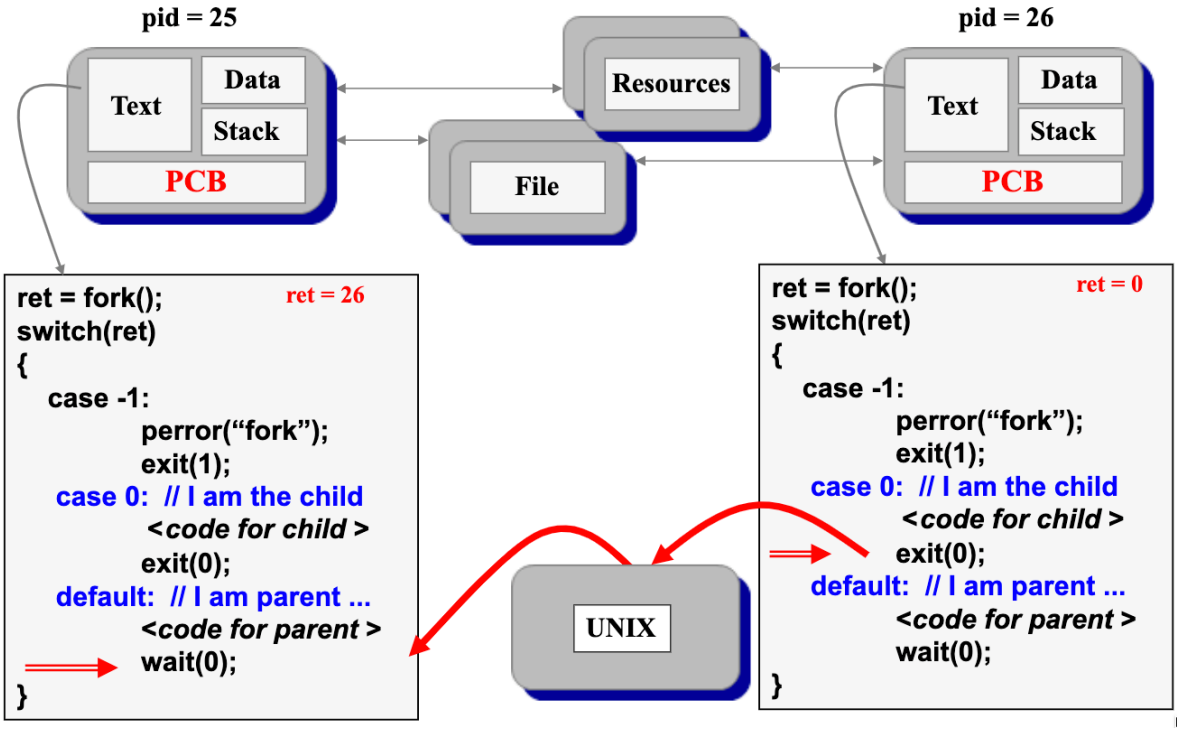
Sem utilizar uma máquina virtual ou algum artifício de distribuição de dados, um arquivo ELF (binário) não pode ser executado no Windows. Explique por quê ?

CRIAÇÃO DE PROCESSOS COM fork() – sys_fork()

- A criação de processos, em diversos sistemas operacionais, funciona com base na chamada de duas funções: **fork()** (no nível usuário) e uma correspondente chamada **sys_fork()** (no nível do kernel do S.O.).



- O mecanismo de funcionamento do fork() é mostrado abaixo:



EXERCÍCIO COM DISCUSSÃO EM DUPLAS

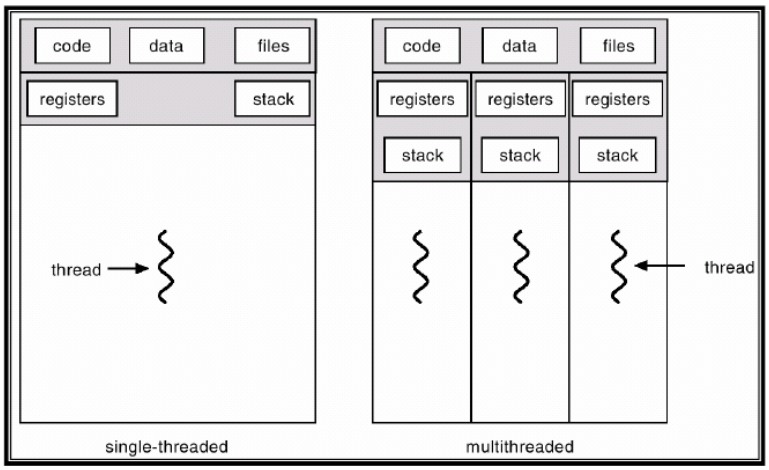
Construa um programa C que tenha dois processos rodando concorrentemente: vi (editor de texto) e sh (interpretador de comandos).

THREADS

- Um dos problemas no lançamento de processos é a necessidade de se criar um novo endereçamento para programa, dados e pilha de execução, o que pode tornar o desempenho do programa bastante lento.

- Ao invés de criarmos um endereçamento novo, pode compartilhar os endereços do pai de forma concorrente. Processos que compartilham os dados do pai são chamado de **processos de peso leve (lightweight process)** ou **threads**.

- Threads sempre estarão vinculadas a um processo. Na rodada de um programa, existirá, pelo menos, uma thread.



- Sistemas operacionais que suportam **somente uma thread** por processo são chamados de **single-threaded**. Caso contrário, de **multithreaded**.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> //Header file for sleep(). man 3 sleep for details.
#include <pthread.h>

void *myThreadFun(void *vargp)
{
    sleep(1);
    return NULL;
}

int main()
{
    pthread_t thread_id;
    printf("Before Thread\n");
    pthread_create(&thread_id, NULL, myThreadFun, NULL);
    pthread_join(thread_id, NULL);
    printf("After Thread\n");
    exit(0);
}
```

EXERCÍCIO EXTRA-CLASSE

Produza um documento para descrever como MINIX cria processos e, se houver, como ele cria também threads.