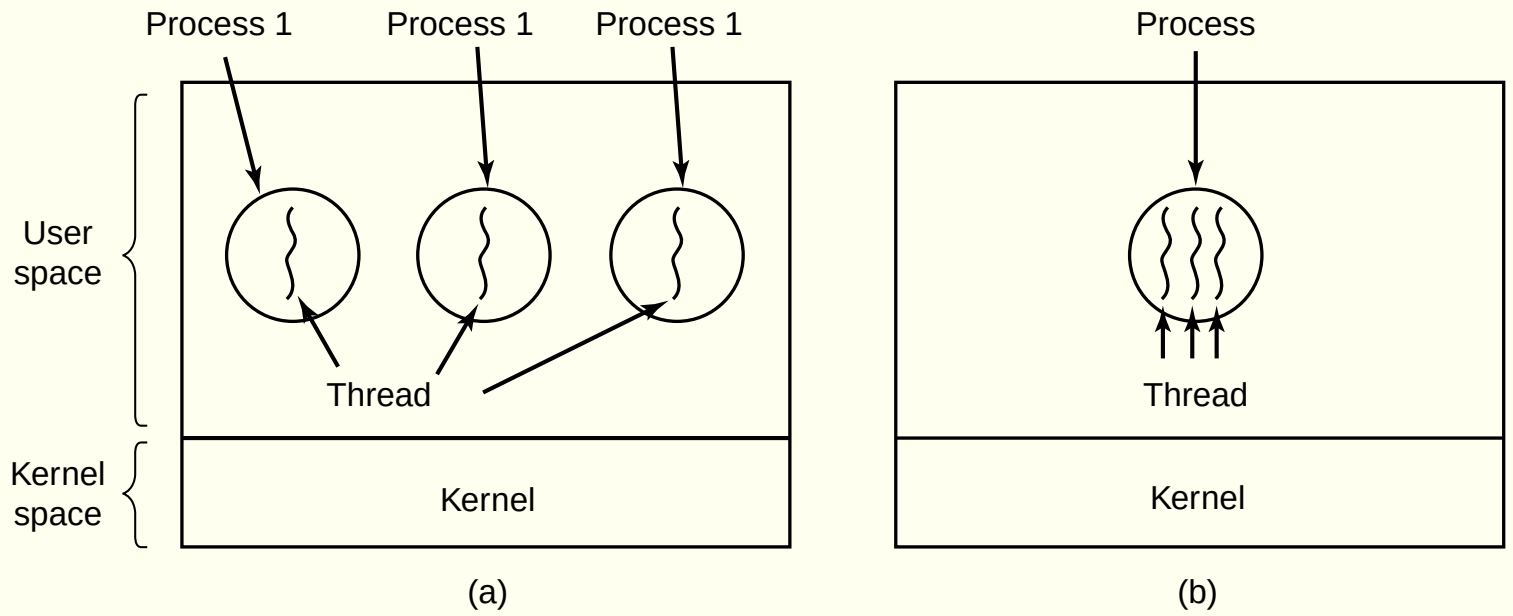
MC514
Sistemas Operacionais:
Teoria e Prática

1s2006

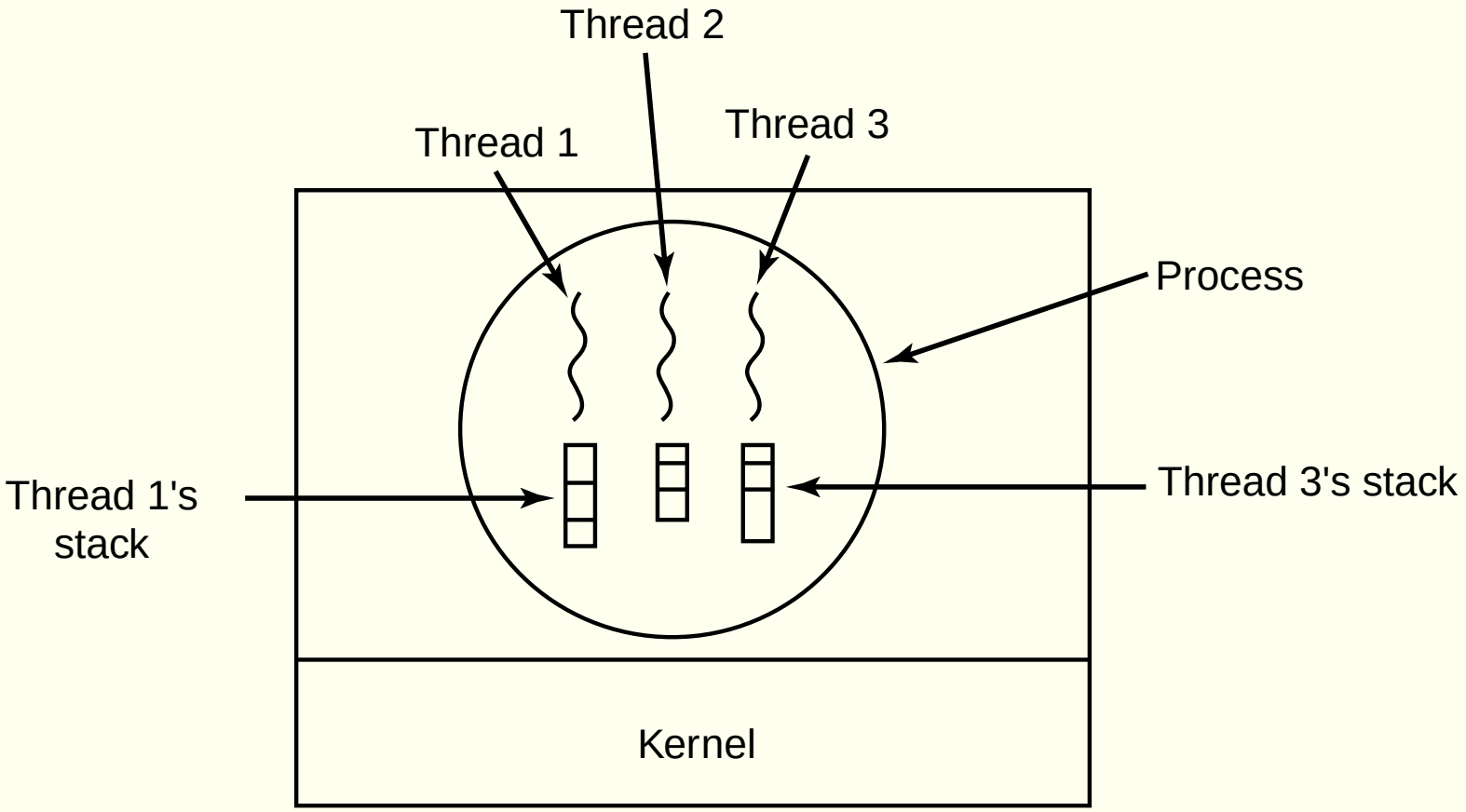
Criação de processos

Processos e threads



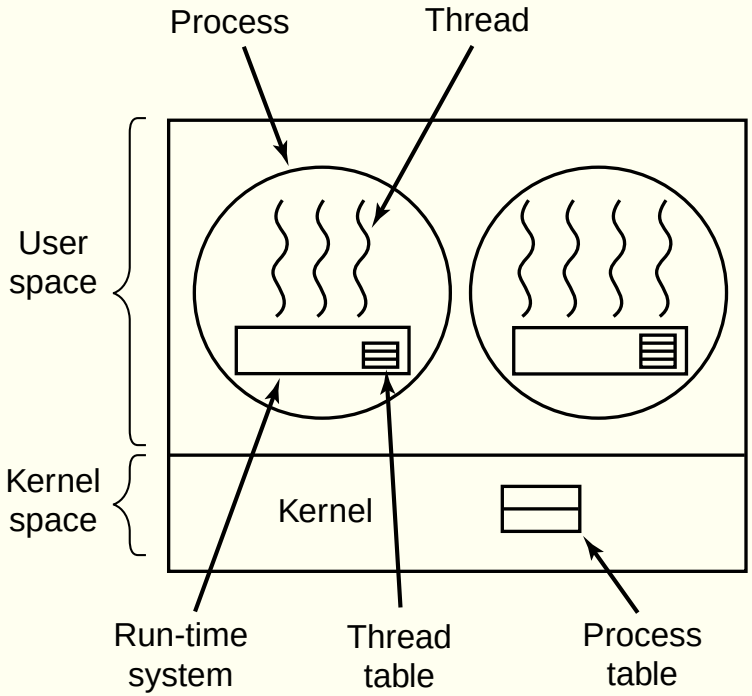
Processos e threads

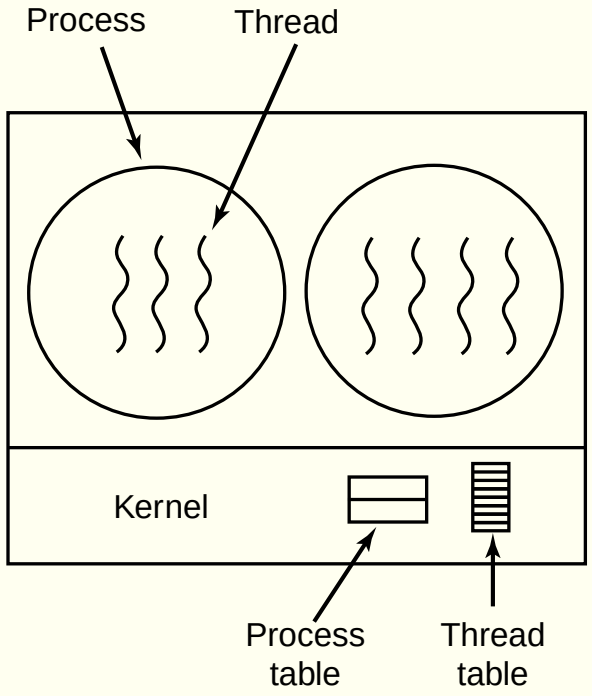
Pilhas independentes



Processos e threads

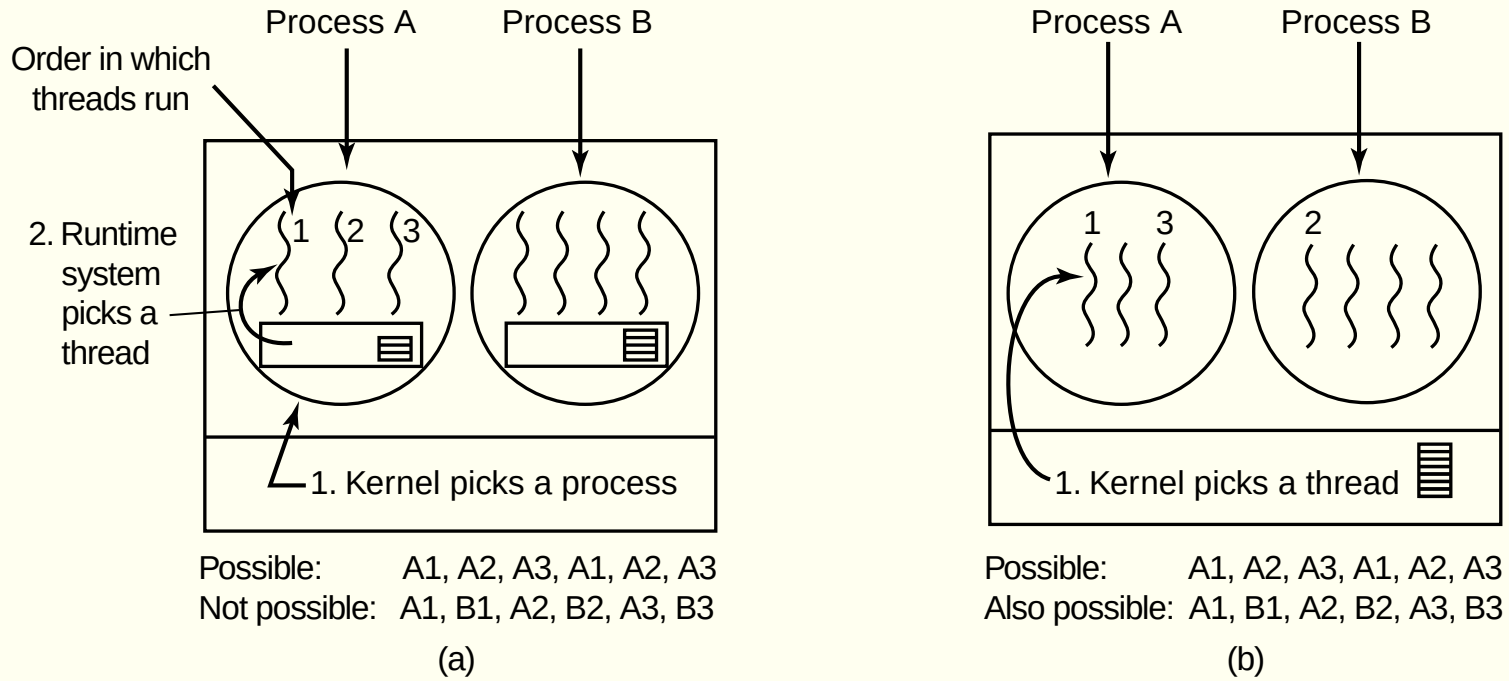
Threads de usuário e de núcleo





Processos e threads

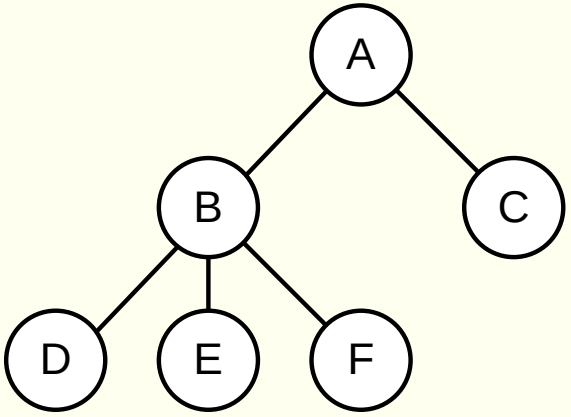
Escalonamento de threads



Criação de processos

- início do sistema
- chamada ao sistema
- requisição do usuário
- início de um job

Hierarquia de processos



Como implementar uma arquitetura como esta utilizando a chamada fork?

fork()

- Cria um novo processo, que executará o mesmo código
- Retorna

  PID do processo criado para o pai e

  0 para o processo filho
- Veja os códigos: fork1.c fork2.c fork3.c

Espaços de endereçamento distintos

```
if (fork() == 0)
  s = 0;
  printf("Processo filho, s = %d\n", s);
} else {
  s = 1;
  printf("Processo pai, s = %d\n", s);
}
```

- Veja o código: fork4.c

wait()

```
pid_t wait(int *status);
```

- Aguarda pela morte de um filho.
- Bloqueia o processamento
- Retorna o pid do filho morto
- `status` indica causa da morte
- Veja os códigos: wait1.c e wait2.c

waitpid()

```
pid_t waitpid(pid_t pid, int *status,
              int options);
```

- Aguarda pela morte de um filho.
  - Específico `pid = PID`
  - Qualquer `pid = -1`
- Versão não bloqueante (options = `WNOHANG`)
- Veja o código waitpid1.c

Argumentos para os processos

- Exemplo

```
$ cp
cp: missing file arguments
Try ‘cp --help’ for more information.
$ cp arq-origem arq-destino
```

- Implementação

```
int main(int argc, char** argv)
```

Variáveis de ambiente

- Exemplo

```
PWD=/l/home/islene/mc514
HOME=/home/islene
LOGNAME=islene
```

- Implementação

```
int main(int argc, char** argv, char** envp)
```

- Veja o código: envp.c

Execução de outros códigos

Família exec

```
int execve(const char *filename,
           char *const argv [],
           char *const envp[]);
```

- Executa o programa `filename`, passando `argv[]` e `envp[]` como argumentos
- Outras opções: `execl()`, `execlp()`, `execle()`, `execv()` e `execvp()`
- Veja o código: execve1.c

Terminação de processos

- saída normal (voluntária)
- saída por erro (voluntária)
- erro fatal (involuntária)
- morto por outro processo (involuntária)
- Veja o código: execve2.c

Shell

```
#define TRUE 1

while (TRUE) {                                  /* repeat forever */
        type_prompt( );                         /* display prompt on the screen */
        read_command(command, parameters);      /* read input from terminal */

        if (fork( ) != 0) {                     /* fork off child process */
                /* Parent code. */
                waitpid(-1, &status, 0);        /* wait for child to exit */
        } else {
                /* Child code. */
                execve(command, parameters, 0); /* execute command */
        }
}
```

Shell

- Como implementar processos em *background*?

```
$ cp arquivo_grande copia_grande &
```

Como criar threads?

- Veja a documentação da função `clone()`