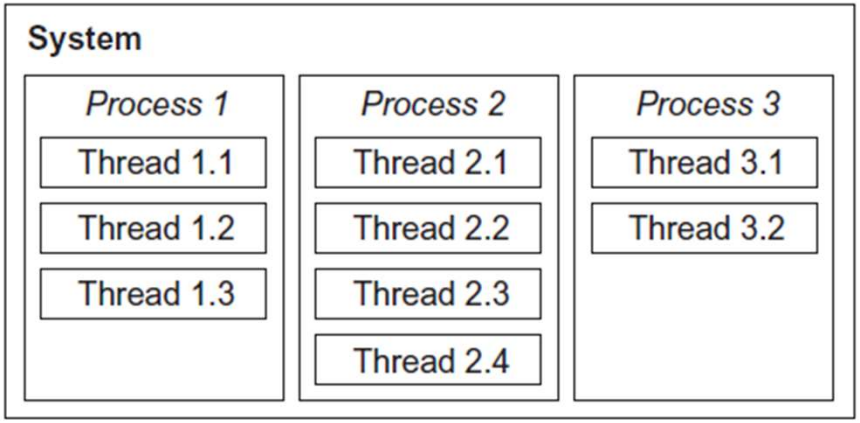
- XXXXXX

Capítulo 4

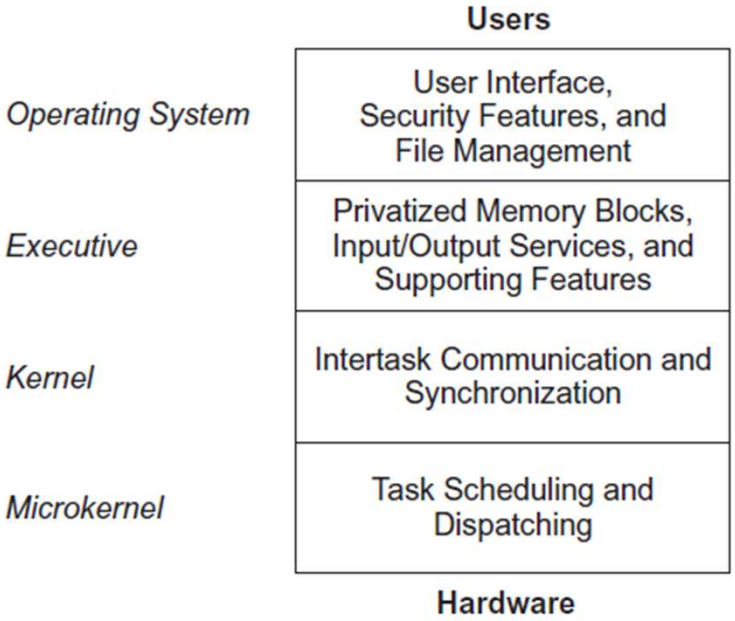
Sistemas operacionais para tempo real[1]

Introdução

- Papel de SO em tempo real:
  - Coordenar concorrência virtual de processos
    - Todo processo tem: ID, nível prioridade e estado execução



  - Gerência memória e Acesso a recursos de hardware



Introdução

- Características de SO para STR:
  - Foco nos processos:
    - escalonador, despacho (bookkeeping), comunicação entre processos e sincronização
  - Kernel mais simples possível
  - Tenta atender a requerimentos de tempo
  - Ambiente de multitasking: robusto e mais flexível

- Exemplos:
  - **FreeRTOS**: para microcontroladores e sistemas embarcados pequenos.
  - **VxWorks**: usado pela indústria aeroespacial e automação complexa
  - **QNX**: para missões críticas.
  - **ThreadX**: usado para eletrônicos de consume, carro, etc.
  - **Nucleus RTOS**: uso diversificado.

Estratégias

- A) Pseudo Kernel:
  - Usando a “polled loop”

```
for(;;) {                    /* do forever */
        if (packet_here)     /* check flag */
          {
          process_data();    /* process data */
          packet_here=0;     /* reset flag */
          }
        }
```

  - Estrutura de código cíclica

```
for(;;) {           /* do forever */
        task_1();
        task_2();
        ...
        task_n();
        }
```

Estratégias

- Co-rotinas:
  - Controle fica alternando entre task_a e task_b

```
void task_a(void)
{
for(;;)
   {
   switch(state_a)
    {
    case 1: phase_a1();
      break;   /* to dispatcher */
    case 2: phase_a2();
      break;   /* to dispatcher */
    case 3: phase_a3();
      break;   /* to dispatcher */
    }
   }
}
void task_b(void)
{
for(;;)
   {
   switch(state_b)
    {
    case 1: phase_b1();
      break;   /* to dispatcher */
    case 2: phase_b2();
      break;   /* to dispatcher */
    case 3: phase_b3();
      break;   /* to dispatcher */
    }
   }
}
```

Estratégias

- B) Sistemas “interrupt-Only”:
  - Usam triggers de interrupções de hardware e/ou software:

```
void main(void)
{
  init();             /* system initialization */
  while(TRUE);        /* jump-to-self */
}
void int_1(void)      /* interrupt handler 1 */
{
  save(context);      /* save context to stack */
  task_1();           /* execute task 1 */
  restore(context);   /* restore context */
}
void int_2(void)      /* interrupt handler 2 */
{
  save(context);      /* save context to stack */
  task_2();           /* execute task 2 */
  restore(context);   /* restore context */
}
void int_3(void)      /* interrupt handler 3 */
{
  save(context);      /* save context to stack */
  task_3();           /* execute task 3 */
  restore(context);   /* restore context */
}
```

Estratégias

- C) Sistemas preempitivos:
  - Escalonamento circular (Round-Robin) por:
    - Prioridades estáticas
    - Prioridades dinâmicas
      - Mecanismo de software de “watchdog”
    - Quanta pré-definida

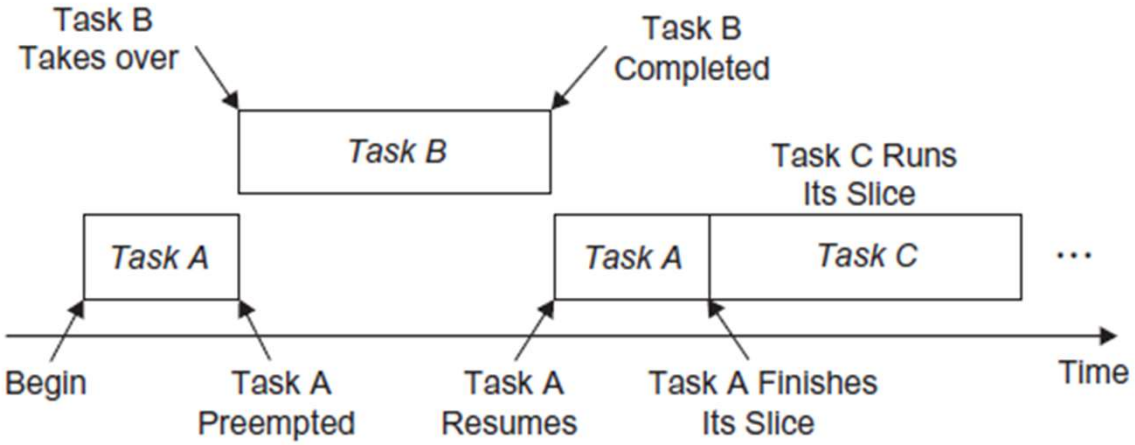


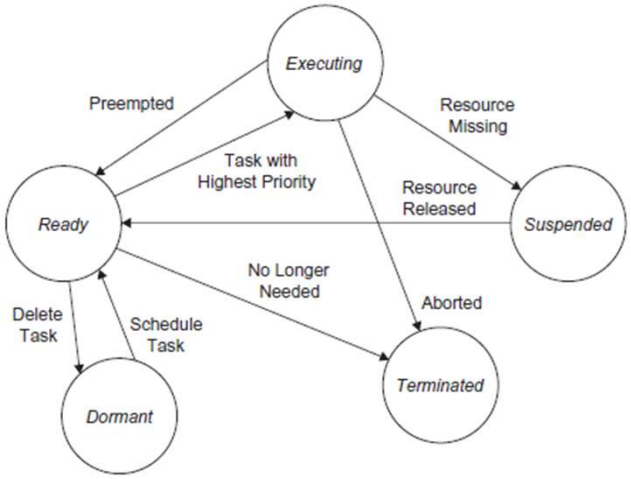
Figure 3.7. Hybrid (round-robin/preemptive) scheduling of three tasks with two priority levels.

Modelo controle STR

- Modelo TCB (‘task control block’)
  - Modelo usado para controlar as task

Estados de escalonamento

- Máquina de estado básica das tasks



- *Precedence constraints:* Specify if any task needs to precede other tasks.
- *Release time* $r_{i,j}$*:* The release time of the jth instance of task τ_i.
- *Phase* ϕ_i*:* The release time of the first instance of task τ_i.
- *Response time:* The time span between the task activation and its completion.
- *Absolute deadline* d_i*:* The instant by which task τ_i must complete.
- *Relative deadline* D_i*:* The maximum allowable response time of task τ_i.
- *Laxity type:* The notion of urgency or leeway in a task's execution.
- *Period* p_i*:* The minimum length of interval between two consecutive release times of task τ_i.
- *Execution time* e_i*:* The maximum amount of time required to complete the execution of task τ_i when it executes alone and has all the resources it needs.

- Deadline task:

$$d_{i,k} = r_{i,k} + p_i = \phi_i + kp_i,$$

- Determinação prioridades

TABLE 3.1. Example Task Set for Frame-Size Calculation

τ_i	p_i	e_i	D_i
τ_1	15	1	15
τ_2	20	2	20
τ_3	22	3	22

Estados de escalonamento

- Teorema RM (rate-monotonic)

> Given a set of periodic tasks and preemptive priority scheduling, then assigning priorities such that the tasks with shorter periods have higher priorities (rate-monotonic), yields an optimal scheduling algorithm.

  - Considerando que o fator U (utilização de processador) é dado por:

$$U \approx n(2^{1/n} - 1).$$

  - A taxa de utilização máxima para n alto é (RMA – *rate monotonic algorithm*):

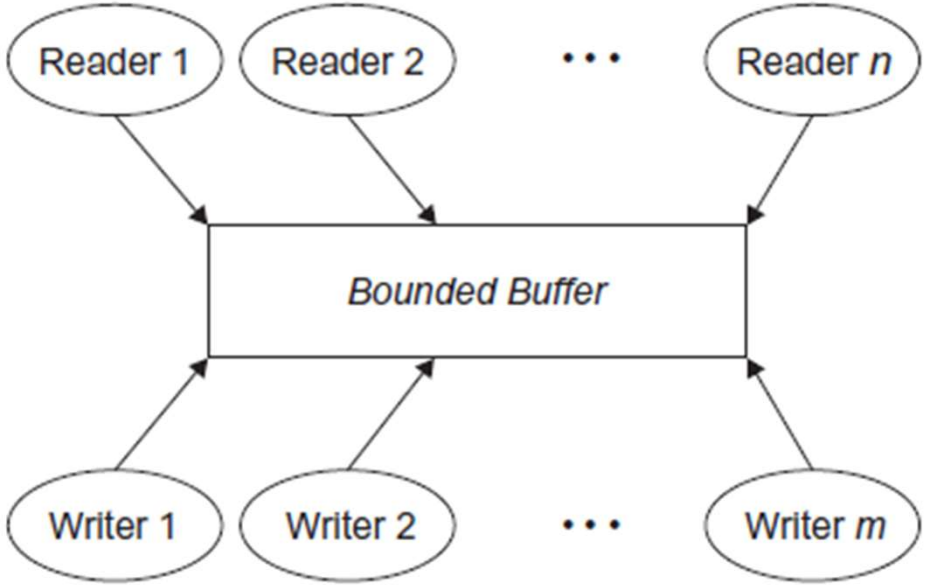
$$\lim_{n\to\infty} n\left(2^{1/n} - 1\right) = \ln 2 \approx 0.69$$

TABLE 3.3. Upper Bounds of the CPU Utilization Factor, U (%), for n Periodic Tasks Scheduled Using the RMA

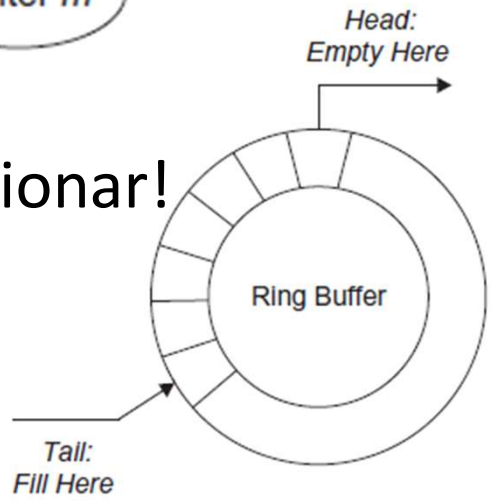
n	1	2	3	4	5	6	...	$\to\infty$
U (%)	100	83	78	76	74	73	...	69

Serviços de OS

- O ‘problema’ de tasks consumidores e produtoras em buffer memórias



  - Buffer circular pode ou não funcionar!



Serviços de OS

- Arbitragem: semáforo
  - Região memória que bloqueia uma região crítica que não pode ser acessada por 2 (ou mais) processos/recursos

```
/* Task_1 */
...
wait(&s);  /* wait until A/D available */
select_channel(acceleration);
a_data=ad_conversion(); /* measure */
signal(&s) /* release A/D */
...

/* Task_2 */
...
wait(&s);  /* wait until A/D available */
select_channel(temperature);
t_data=ad_conversion(); /* measure */
signal(&s) /* release A/D */
...
```

Serviços de OS

- Serviço de timers e clock
  - Timers de hardware e de software
  - “incerteza do ‘delay’ “
- Gerência de memória
  - Coleta, bloqueio, swapping, paging, etc

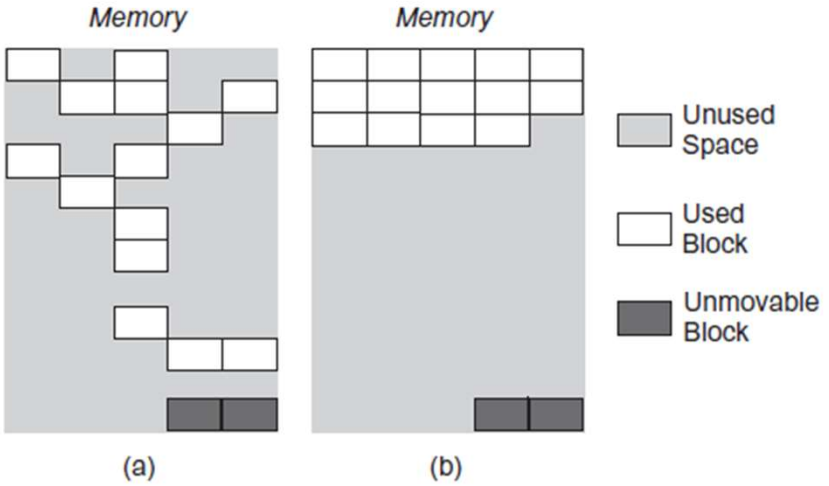


Figure 3.21. Fragmented memory (a) before and (b) after compaction; the unmovable blocks represent the root program of the real-time operating system.

Critérios escolha

- Tipo de sistema STR: hard, firm ou soft
  - Tolerância ao erro
  - Previsibilidade
  - Estabilidade em pico de cargas
  - Tempo de desenvolvimento (suporte)
  - Capacidade de manutenção

- Algumas métricas 'm_i' ($m_i \in [0, 1]$) para STR hard com pesos w_i.

$$\bar{M} = \frac{1}{13} \sum_{i=1}^{13} w_i m_i$$

Criterion	Description	Rating	Comment
m_1	Minimum interrupt latency	*	CPU dependent
m_2	Maximum number of tasks	0.5	32 task-priority levels
m_3	Total memory required	0.7	ROM: 60 K bytes
m_4	Scheduling mechanism	0.25	Preemptive only
m_5	Communicate/synchronize	0.5	Direct message passing
m_6	After-sale support	0.5	Paid phone support
m_7	Application availability	1	Various
m_8	CPUs supported	0.8	Various
m_9	Source code	1	Available
m_{10}	Save the context	*	Unknown
m_{11}	Cost	0.5	$2500 + royalty fee
m_{12}	Development platforms	*	Unknown
m_{13}	Networks and protocols	1	Various

M1 = latência interrupção; **M2** = número máximo tasks SO; **M3** = memória requerida para SO; **M4** = mecanismo escalonamento; **M5** = recursos sincronismo (mutex, semáfaro, etc), **M6** = 'after-sale', **M7** = disponibilidades soft desenvolvimento, **M8** = suporte CPUs, **M9** = troca de contexto entre tarefas, **M10** = suporte protocolos (rede, principalmente) .....

Referências

- [1] Real-time systems design and analysis – tools for the practitioner