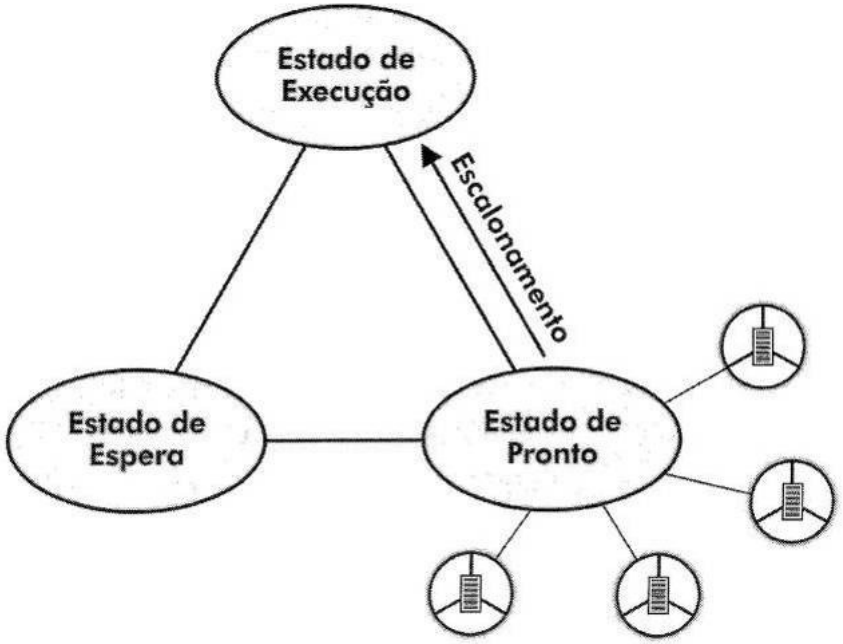
Gerência do Processador
Escalonador de Tarefas

Prof. Edwar Saliba Júnior

Maio de 2011

Introdução

- A gerência do processador é uma das atividades mais importantes em sistemas multitarefas/multi-programáveis.



Funções Básicas

- Manter o processador ocupado a maior parte do tempo;
- Balancear o uso da CPU entre processos;
- Privilegiar a execução de aplicações críticas;
- Maximizar o *troughput* do sistema;
- Oferecer tempos de resposta razoáveis para usuários interativos.

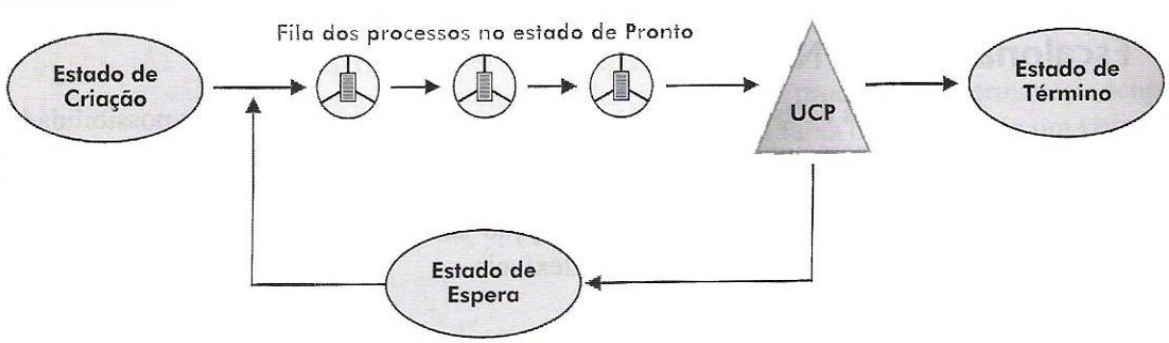
Critérios de Escalonamento

- ***Throughput***: representa o número de processos executados em um determinado intervalo de tempo;
- **Tempo de Processador**: é o tempo que um processo leva no estado de execução durante seu processamento;
- **Tempo de Espera**: é o tempo total que um processo permanece na fila de pronto durante seu processamento, aguardando para ser executado;
- **Tempo de *Turnaround***: é o tempo que um processo leva desde a sua criação até ao seu término;
- **Tempo de Resposta**: é o tempo decorrido entre uma requisição ao sistema ou a aplicação e o

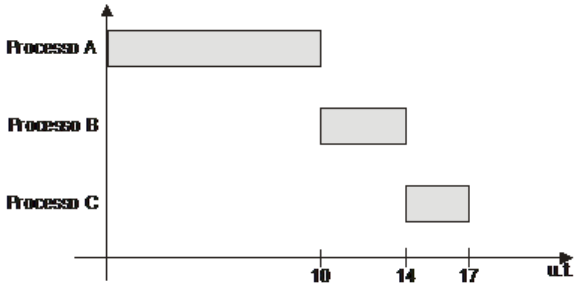
Tipos de Escalonamento

- **Não Preemptivos**: foi o primeiro tipo de escalonamento implementado nos sistemas multiprogramáveis, onde predominava tipicamente o processamento em *batch*;
- **Preemptivos**: caracterizado pela possibilidade do sistema operacinal interromper um processo em execução e passá-lo para o estado

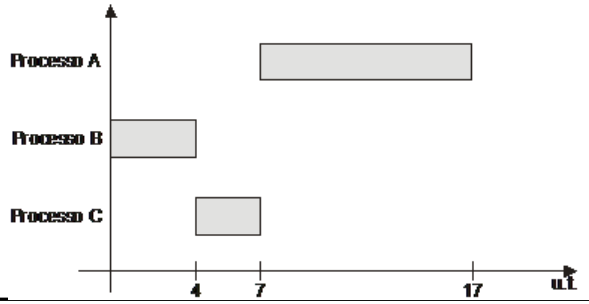
Alguns dos Principais Algoritmos de Escalonomento



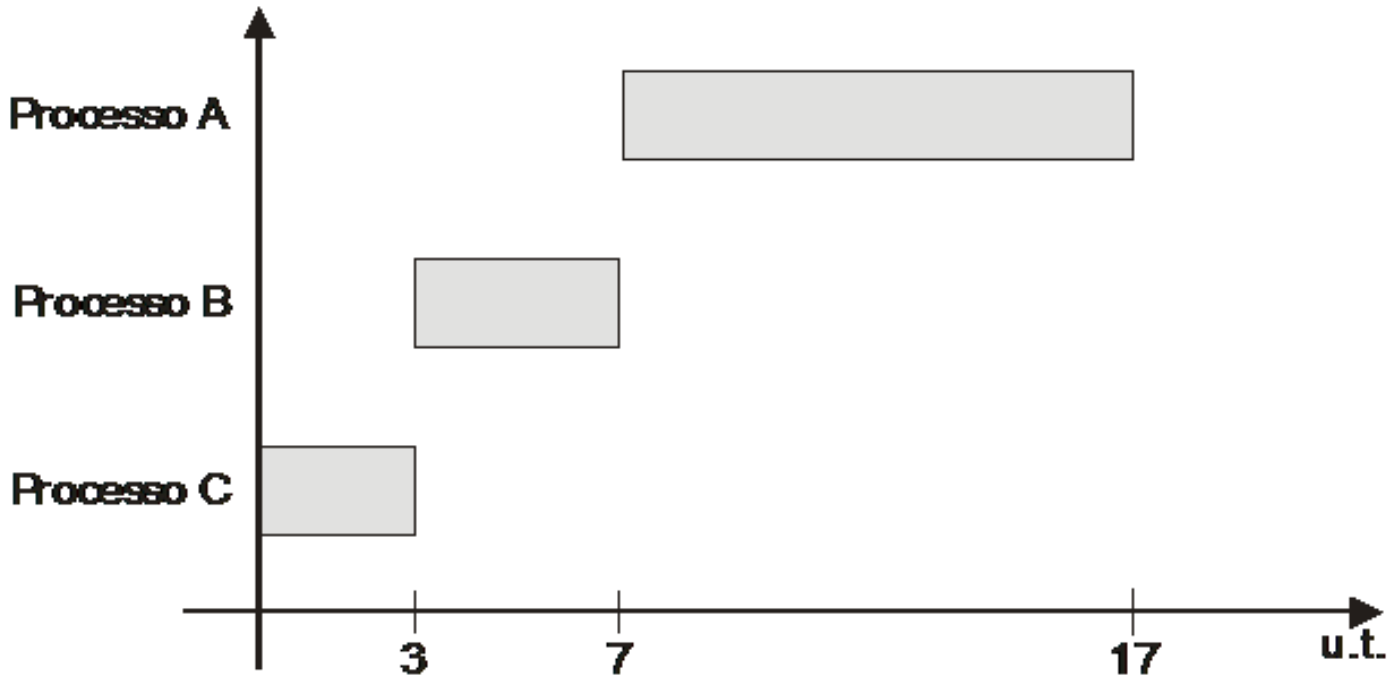
First in first out (FIFO)



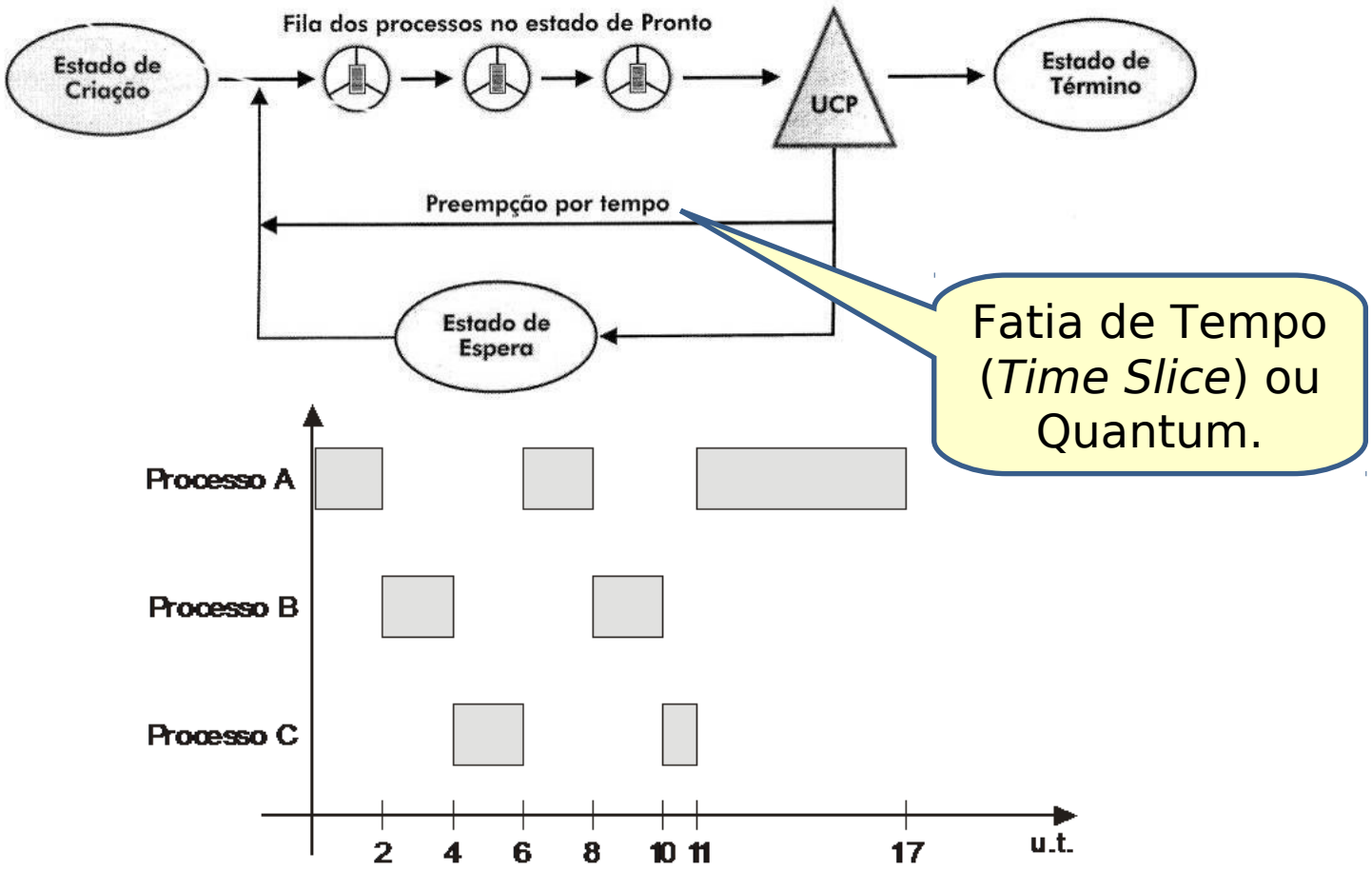
Processo	Tempo de processador (u.t.)
A	10
B	4
C	3



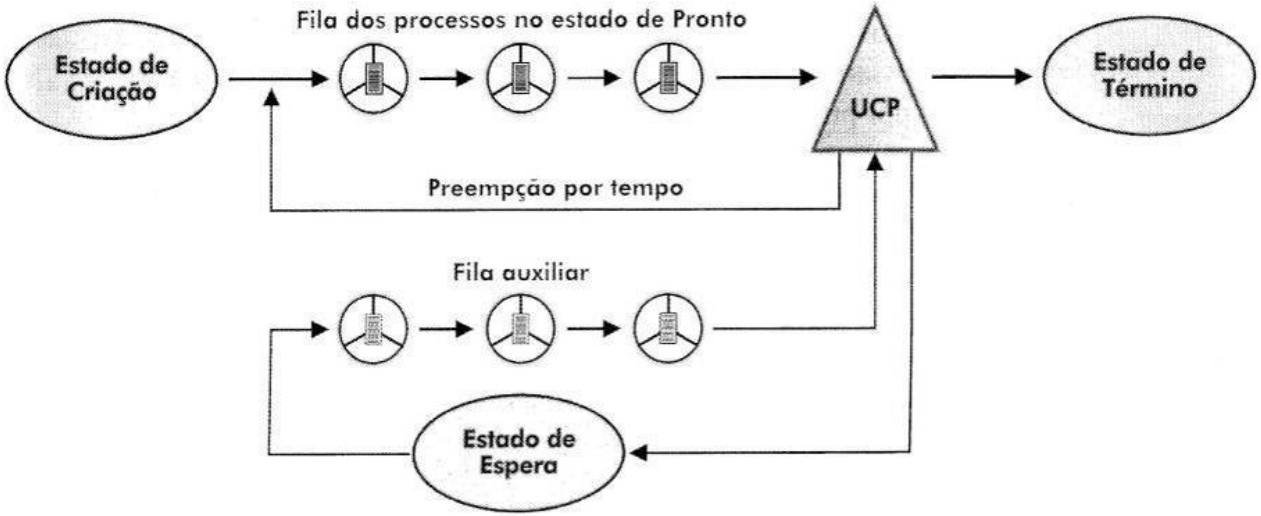
Shortest Job First (SJF)



Round Robin (RR) ou Esc. Circular

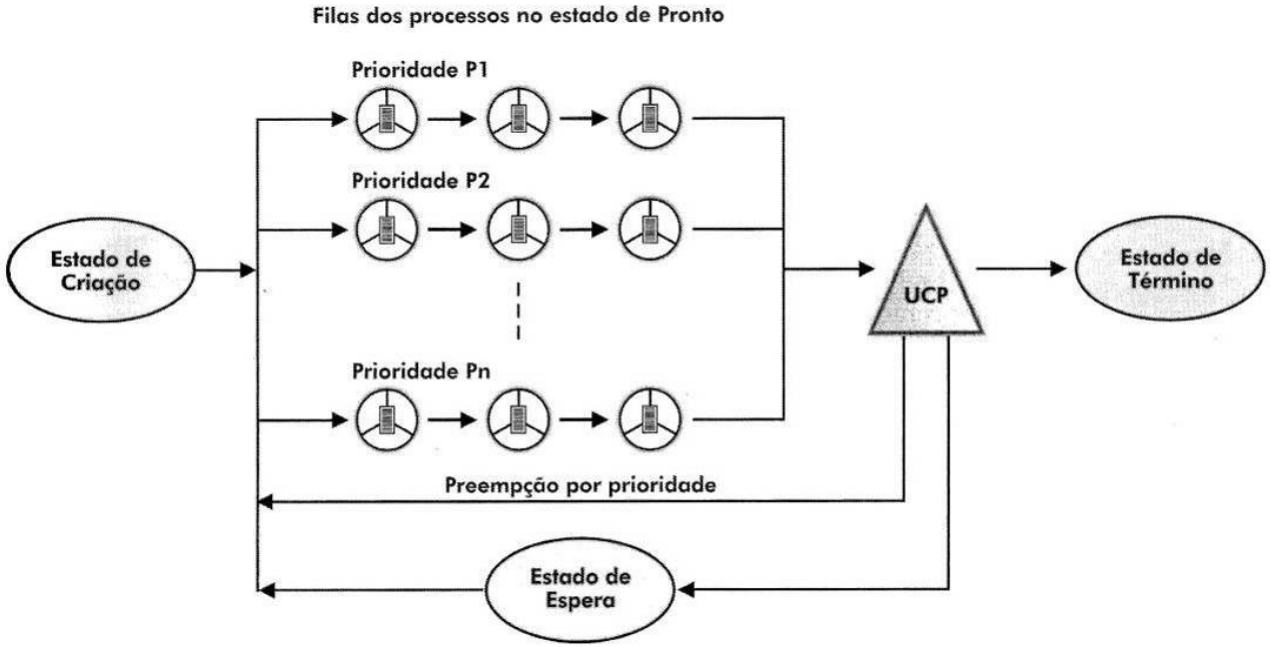


Escalonamento Circular Virtual



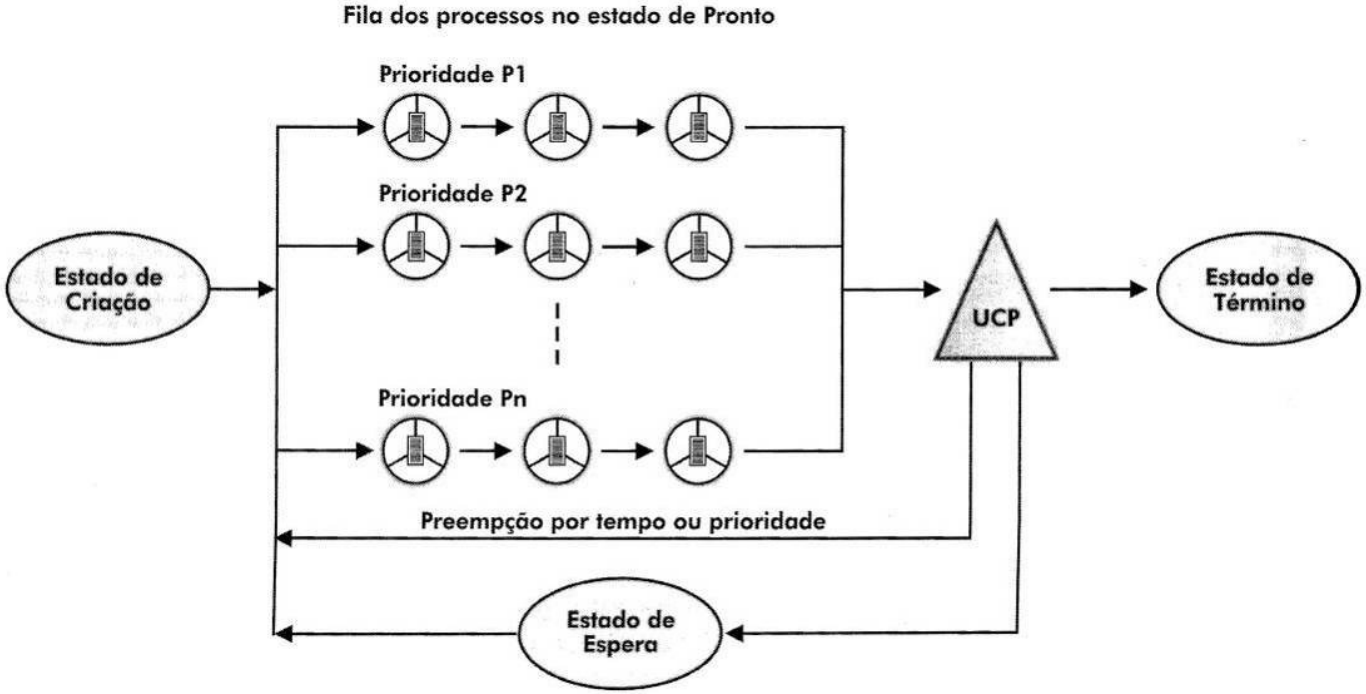
- Um problema presente no Escalonomento Circular é que, devido às suas características, os processos CPU-*bound* tendem a utilizar integralmente sua fatia de tempo, enquanto os processos I/O-*bound* têm mais chances de passar para o estado de espera antes de sofrerem preempção por tempo.

Escalonamento por Prioridade



- Escalonamento preemptivo realizado com base em um valor associado a cada processo, denominado prioridade de execução;
- A perda do uso do processador só ocorrerá no caso de uma mudança voluntária para o estado de espera ou quando um processo de prioridade maior passa para o estado de pronto.

Esc. Circular com Prioridade



- Implementa o conceito de fatia de tempo e de prioridade de execução associada a cada processo;
- Um processo permanece no estado de execução até que termine seu processamento, voluntariamente passe para o estado de espera ou sofra uma preempção por tempo ou prioridade.

Proposta

- Escolher e construir, usando linguagem C, um escalonador de processos.

Linguagem C

- Revisão:
  - Estrutura sequencial;
  - Estrutura condicional;
  - Estrutura de exclusão múltipla;
  - Estruturas de repetição;
  - TAD (Tipo Abstrato de Dados) e
  - Vetor.

Estrutura Sequencial

```c
#include <stdio.h>

void main(){
    int v1, v2, v3, res;

    // Leitura de valores do teclado.
    printf("Digite o primeiro valor: ");
    scanf("%d",&v1);
    printf("Digite o segundo valor: ");
    scanf("%d",&v2);

    // Atribuição.
    v3 = 100;

    res = v1 + v2 + v3;

    printf("\n\nO total da soma e: %d \n\n", res);
}
```

Estrutura Condicional

```c
#include <stdio.h>

void main(){
    float v1, v2, v3, media;

    printf("Digite o valor da primeira nota: ");
    scanf("%f",&v1);
    printf("Digite o valor da segunda nota: ");
    scanf("%f",&v2);
    printf("Digite o valor da terceira nota: ");
    scanf("%f",&v3);

    media = (v1 + v2 + v3) / 3;

    if(media >= 6){
        printf("\n\nAluno Aprovado!\n\n");
    }
    else{
        printf("\n\nAluno Reprovado!\n\n");
    }
}
```

Estrutura de Exclusão Múltipla

```c
#include <stdio.h>

void main(){
    int dia = 0;

    printf("Digite o número do dia: ");
    scanf("%d",&dia);

    switch(dia){
        case 1:
            printf("\n\nDomingo");
            break;
        case 2:
            printf("\n\nSegunda-feira");
            break;
        case 3:
            printf("\n\nTerça-feira");
            break;
        case 4:
            printf("\n\nQuarta-feira");
            break;
        case 5:
            printf("\n\nQuinta-feira");
            break;
        case 6:
            printf("\n\nSexta-feira");
            break;
        case 7:
            printf("\n\nSábado");
            break;
        default:
            printf("\n\nValor inválido!");

    }
}
```

Estruturas de Repetição

```c
#include <stdio.h>

void main(){
    int val = 0;

    while(val < 100){
        printf("\nValor: %d", val);
        val = val + 1;
    }
}
```

```c
#include <stdio.h>

void main(){
    int val = 0;

    do{
        printf("\nValor: %d", val);
        val = val + 1;
    }while(val < 100);
}
```

```c
#include <stdio.h>

void main(){
    int val;

    for(val = 0; val < 100; val++){
        printf("\nValor: %d", val);
    }
}
```

Tipo Abstrato de Dados (TAD)

```c
#include <stdio.h>

typedef struct Pessoa{
    int codigo;
    char nome[50];
    char identidade[10];
};

void main(){
    struct Pessoa p1, p2;

    printf("Digite o código da 1a. pessoa: ");
    scanf("%d",&p1.codigo);
    fflush(stdin);
    printf("Digite o nome da 1a. pessoa: ");
    gets(p1.nome);
    printf("Digite a identidade da 1a. pessoa: ");
    gets(p1.identidade);

    printf("\n\nDigite o código da 2a. pessoa: ");
    scanf("%d",&p2.codigo);
    fflush(stdin);
    printf("Digite o nome da 2a. pessoa: ");
    gets(p2.nome);
    printf("Digite a identidade da 2a. pessoa: ");
    gets(p2.identidade);

    printf("\n\nDados das Pessoas\n");
    printf("=================\n");

    printf("\nCódigo da 1a. pessoa: %d", p1.codigo);
    printf("\nNome da 1a. pessoa: %s", p1.nome);
    printf("\nIdentidade da 1a. pessoa: %s\n\n", p1.identidade);

    printf("\nCódigo da 2a. pessoa: %d", p2.codigo);
    printf("\nNome da 2a. pessoa: %s", p2.nome);
    printf("\nIdentidade da 2a. pessoa: %s\n\n", p2.identidade);
}
```

Vetor

```c
#include <stdio.h>

void main(){
    int vet[100],
        cont = 0, i;

    while(cont < 100){
        vet[cont] = cont * 10;
        cont++;
    }

    for(i = 0; i < 100; i++){
        printf("\nPosição %d do vetor: %d", i, vet[i]);
    }
}
```

Revisão Finalizada!

- Mãos a obra!!!!

Bibliografia

- MACHADO, F. B.; MAIA, L. P. **Arquitetura de Sistemas Operacionais,** 3ª Ed., Rio de Janeiro: LTC Editora, 2002.