

# **Simulador de Índice de Massa Corporal (IMC)**

Victor Dallagnol Bento

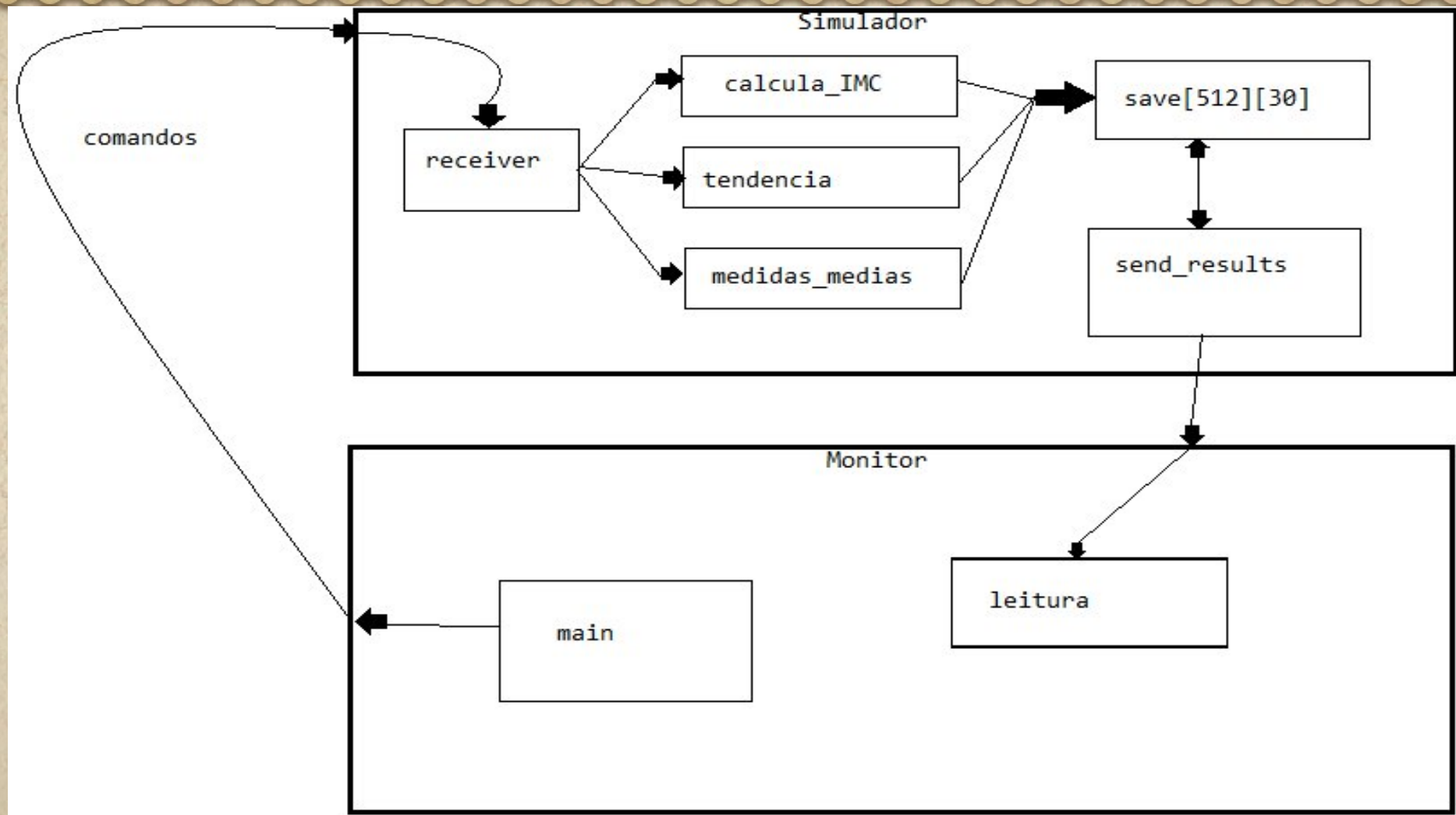
Sistemas Operacionais em Tempo Real

Universidade federal de Santa Maria

Julho de 2018



# Ideia Principal





# Tarefas Periódicas

❖ *void \*calcula\_IMC(void \*arg):*

Efetua o cálculo do IMC se altura e peso não forem nulos.


❖ *void \*tendencia(void \*arg):*

Verifica se indivíduo está propenso a desnutrição, sobrepeso, obesidade ou se está nos níveis normais de IMC.

❖ *void \*medidas\_medias(void \*arg):*

Efetua a média de todos os pesos e alturas registrados até o momento.





Todas as tarefas periódicas salvam seus resultados em variáveis globais, entretanto, somente a função que terá seu comando especificado salvará seu resultado no buffer de informações *save[256][5]*. Se o comando não for condizente com o especificado, os cálculos são efetuados e as variáveis são atualizadas.

Sempre que uma função escreve no buffer, a variável *linha*, que indica a linha do buffer, é atualizada;





# Comunicação em Rede

## ❖ Envio de Dados (*monitor* - main):

```
printf("<?>\tInforme os comandos:\n");
do {
    bzero(buffer, sizeof(buffer)); /* Limpa buffer */
    fgets(buffer, 50, stdin);
    if (strcmp(buffer, "exit\n") == 0) {
        break;
    }
    /* Condição para comandoa aceitos */
    if (buffer[0] == '0' || buffer[0] == '1' || buffer[0] == '2'){
        n = send(sockfd, buffer, 50, 0);
        if (n == -1) {
            printf("Erro escrevendo no socket!\n");
            return -1;
        }
    } else {
        printf("\n<?>\tComando: %s Inválido, informe os dados novamente.\n\n", buffer);
    }
} while (1);
```

Envia Comandos válidos para o simulador:

**0** peso altura - (calculo do IMC);

**1** peso altura - (médias);

**2** peso altura - (tendência do indivíduo).

Caso os comandos não forem válidos, espera por comandos válidos.

Não possui *mutex* pois possui apenas um cliente/monitor.



## ❖ Captura dos Dados (*simulador* - receiver):

```
void *receiver(void *arg) { /* Recebe dados
    int n, ctrl;
    char buffer[T_buffers], *aux;
    while (1) {
        bzero(buffer, sizeof(buffer)); /* zera bu
        n = read(newsockfd, buffer, 50);

        if (n <= 0) { /* condição para
            printf("Erro lendo do socket!\n");
            exit(1);
        }
    }
}
```

↓  
*Função que recebe dados,  
limpa o buffer para depois  
escrever nele.*

```
pthread_mutex_lock(&protect);
ctrl = 0;
aux = strtok(buffer, " ");
id_comando[id_co_peso_alt] = atoi(aux);
while (aux != NULL) {
    aux = strtok(NULL, " ");
    if (ctrl == 0) { // Controle para
        altura[id_co_peso_alt] = atof(aux);
        ctrl = 1;
    } else if (ctrl == 1) { // Controle para
        peso[id_co_peso_alt] = atof(aux);
        ctrl = 2; // Não salva ma
    }
}
if (id_co_peso_alt == T_buffers) /* Co
    id_co_peso_alt = 0;
else
    id_co_peso_alt++;
pthread_mutex_unlock(&protect);
}
```

→ *Conversão dos  
dados recebidos e  
atualização das  
variáveis globais -  
vetores (uso do  
mutex protect)*



## ❖ Envio dos Resultados (*simulador* - *send\_results*):

```
pthread_mutex_lock(&mutex);  
n = write(newsockfd,buffer,50);  
if (n < 0) {  
    printf("Erro escrevendo no socket!\n");  
    exit(1);  
}  
printf("<!\>\tMensagem enviada\n");  
pthread_mutex_unlock(&mutex);  
}
```

A função *send\_results* é responsável pela comunicação com o monitor.

Nela, o buffer *save* é percorrido (linha á linha) até que o comando específico seja encontrado. Após a procura, se o comando for encontrado, a flag *find* é atualizada e o resultado é enviado ao monitor.

Após o envio a linha é inutilizada (comando inválido é escrito na mesma) pra que posteriormente possa ser sobreescrita.

Se todos os comandos forem enviados o buffer é considerado vazio (flag *vazio* é atualizada).



## ❖ Captura de Dados (*monitor* - leitura):

```
while (1) {
    bzero(buffer, sizeof(buffer));
    n = recv(sockfd, buffer, 50, 0);

    if (n <= 0) {
        printf("Erro lendo do socket!\n");
        exit(1);
    }

    /*
     * Identificação dos comandos para amostragem de dados:
     * id_comand recebe o comando.
     * imc recebe o calculo do IMC.
     */
    if (buffer[0] == '0'){
        id_comand = strtok(buffer, " ");
        imc = strtok(NULL, " ");

        if(imc < "18.5")
            printf("<!\>\tIMC: %s kg/m^2 - Magreza\n", imc);
        else if (imc >= "18.5" || imc <= "24.5")
            printf("<!\>\tIMC: %s kg/m^2 - Normal\n", imc);
        else if (imc > "24.9" || imc <= "30")
            printf("<!\>\tIMC: %s kg/m^2 - Sobrepeso\n", imc);
        else
            printf("<!\>\tIMC: %s kg/m^2 - Obesidade\n", imc);
    }
}
```



```
/**
 * _peso_altura recebe comando, e passa para variável id_
 * ctrl_print, serve para separar as informações na hora d
 * mostrando o peso médio (ctrl = 0), depois mostrando a a
 * ao fim a flag ctrl é setada para 0 para que a conversão
 */
} else if (buffer[0] == '1'){
    _peso_altura = strtok(buffer, " ");
    id_comand = _peso_altura;
    ctrl_print = 0;

    while (_peso_altura != NULL) {
        _peso_altura = strtok(NULL, " ");
        if(ctrl_print == 0){
            printf("<!\>\tPeso Médio: %s\n", _peso_altura);
            ctrl_print = 1;
        } else if (ctrl_print == 1) {
            printf("<!\>\tAltura Média: %s\n", _peso_altura);
            ctrl_print = 2; // Não printa mais
        }
    }

    /**
     * Se o comando for 2, será amostrada a tendência do indi
     */
} else{
    id_comand = strtok(buffer, " ");
    tend = strtok(NULL, " ");
    printf("<!\>\tTendencia do individuo: %s\n", tend);
}
}
```



Recebe do *socket* o resultado.

Dependendo do comando existe um print específico. Ocorre uma separação da string recebida em variáveis para serem amostradas.



# Mutex e Variável de Condição

- ❖ No trabalho foram utilizados dois mutex:
  - ***mutex***: Utilizado para estabelecer comunicação com monitor e leitura do socket.
  - ***protect***: Utilizado para atualização das variáveis globais.
- ❖ E uma variável de condição:
  - ***cond***: Utilizada na espera do buffer. Enquanto estiver vazio.



# Problemas e Dificuldades

- ❖ Sincronização das variáveis globais.
- ❖ Conversão de tipos (char → float → char).
- ❖ Escrita e Leitura no buffer de informações.