Experimento #2

Sistemas Operacionais A

**Introdução:**

O experimento realizado envolve a comunicação de processos por meio da transferência de mensagens.

Na primeira parte, o programa realiza a comunicação entre dois processos filhos, que trocam dados para que calculem o intervalo de tempo para o envio e recebimento de 500 mensagens entre si, por meio de uma única fila, sendo o filho 1 o receptor/exibidor dos cálculos e o 2 o emissor das mensagens.

Na segunda parte, o programa deve ser modificado de forma que haja uma segunda fila de mensagens que permita a comunicação entre o filho 1 e um terceiro filho criado, agora o filho receptor não irá exibir os cálculos, mas sim os enviar ao terceiro filho e este será encarregado da exibição.

**Apresentação Dos Erros de Sintaxe e/ou Lógica do programa Exemplo:**

* **Erros de Sintaxe**
  + Linhas 75, 77 e 78, na inclusão das bibliotecas estão faltando o prefixo “sys/”.





* + Linha 154: NO\_OF\_CHILDREN não foi definida anteriormente. E ainda a variação de count foi feita incorretamente e incompleta, sendo certo count++.



* + Linha 156: fork não é chamado como função, e seu valor não é assimilado a rtn, o correto seria rtn = fork();



* + Linha 158: a chamada da função exit() está incorreta, pois não é permitido valor nulo.



* + Linha 173: o conteúdo da chamada printf() não tem o fechamento das aspas.



* + Linhas 190 e 191: chamada incompleta da função wait(), pois está sem a declaração do valor de status.



* + Linha 258: comentário não comentado



* + Linha 296: stdout não é parâmetro de printf e sim de fprintf, para solução pode se transformar em printf(“..”), para manter o modelo da linha 295.



* **Erros de Lógica:**
  + Linha 122: rtn é declarada, porém não inicializado antes de ser utilizado no código



* + Linha 123: count é inicializada sendo q seu valor é substituído no for() da linha

154



* + Linha 155: Condição do if é ilegível, sendo o mais lógico rtn != 0;



* + Linha 158: caso corrigida, a chamada da função exit() faria com que todos os filhos fossem finalizados antes da sua execução.



* + Linha 177: if não realiza o a comparação com o valor de rtn a zero.



* + Linha 196: o código pode ser finalizado caso o retorno a chamada de msgctl() seja diferente de zero.



* + Linha 244: varíavel max não inicializada antes do seu uso.



* + Linhas 295 e 296: número de casas decimais não é suficiente para a análise dos dados, o ideal seria 8 casas depois da vírgula.

****

* + Linha 263: A ordem de incrementação de count está erra, pois incrementa antes de cada interação do for(), seria correto count++;

****

* + Linha 280: delta realiza um decremento e não inicialização com deveria ser feito.



* + Linha 281: atribuição errônea dos cálculos, deveria ser um incremento.



* + Linha 282: o valor de delta é negativado, resultando num valor de tempo negativo.



* + Linha 287: o sinal de comparação do if está invertido, pois se o valor de delta for menor que max, delta deve receber max e não o contrário.



**Respostas às perguntas:**

* **Contidas no texto do Experimento:**
  + **1 - Esclarecer o que são: Berkeley Unix, System V, POSIX, AT & T, *socket*, fila de mensagem, memória compartilhada e *pipes*.**

**Berkeley Unix:** é um sistema operacionalderivado do UNIX da AT&T, distribuído pela Universidade da Califórnia, Berkeley.

**System V ou SysV:** foi uma das primeiras versões comerciais do UNIX, desenvolvido pela American Telephone e Telegraph (AT&T) lançado em 1983, existindo 4 versões ao todo lançadas.

**POSIX (Portable Operating System Interface):** é a familia de normas definidas pela IEEE (em específico IEEE 1003) para a manutenção de compatibilidade dos sistemas operacionais. Tem como objetivo a garantia da portabilidade do código finte de um programa de um sistema operacional (Atendente das Normas POSIX) para outro sistema, sendo as regras usadas como Interface dos sistemas distintos.

**AT&T (American Telephone e Telegraph):** é uma companhia americana que fornece serviços de telecomunicação de vídeo, voz e o mais importante para esse experimento, dados.

**Socket:** também conhecido como Two-way communication, é o provedor de comunicação entre dois processos pertencentes da mesma máquina (Unix Socket) ou na mesma rede (TCP/IP Sockets).

**Fila de Mensagem:** é a estrutura de dados que recebe e armazenas as mensagens até seu processamento ou exclusão.

**Memória Compartilhada:** é a memória que pode ser acessada por vários processos simultaneamente, para permitir a comunicação entre os mesmos ou até para evitar cópias redundantes**.**

**Pipes:** constituem um canal de comunicação unidirecional entre processos com um ascendente comum, como pai e filho. Ao estabelecer um pipe entre os processos um deles pode enviar mensagens ao outro.

* + **2 - As chamadas ipcs e ipcrm apresentam informações sobre quais tipos de recursos?**

Os recursos citados são os IPC’s ou recursos de comunicação entre processos, sendo eles semáforos, filas de mensagens e memória compartilhada.

A chamada ipcs lista esses recursos (ativos no instante chamado) que realizam comunicação entre processos. Já a ipcrm não apresenta nenhuma informação sobre os recursos, porém remove os mesmos quando os processos já finalizaram seu uso.

* + **3 - Qual a diferença entre o *handle* devolvido pela chamada msgget e a chave única?**

A diferença se dá no fato de que o handle devolvido pela chamada msgget é a tradução, feita pelo msgget, da chave única criada estabelecida pelo programador em sua chamada, no campo key.

* + **4 - Há tamanhos máximos para uma mensagem? Quais?**

O tamanho máximo padrão é 4 MB, mas pode ser estendida até 100MB de memória.

* + **5 - Há tamanhos máximos para uma fila de mensagens? Quais?**
  + **6 - Para que serve um typedef?**

O typedef serve para definir ou redefinir um tipo de dado na linguagem C, servindo como “apelido” de uma variável, mas geralmente é utilizada na declaração de “structs”, pois simplifica a declaração de variáveis e parâmetros das estruturas.

* + **7 - Onde deve ser usado o que é definido através de um typedef?**

Este pode ser usado no decorrer todo do código, como um tipo padrão da linguagem C.

* + **8 - Na chamada msgsnd há o uso de *cast*, porém agora utiliza-se um “&” antes de message\_buffer. Explicar para que serve o “&” e o que ocorreria se este fosse removido.**

O “&” serve para que na chamada de msgsnd, o endereço que consta a mensagem armazenada seja acessado para que a mensagem possa ser enviada com êxito, caso o & fosse removido, o código apresentaria um erro de execução, pois o tipo da variável é diferente da solicitada por ser ponteiro, ainda caso o tipo seja aceito e o código não apresente erros, a mensagem seria alterada somente na função na qual foi chamada, não mantendo seu valor atualizado ao fim da mesma.

* **Contidas no código fonte:**
  + **Pergunta 1:** O que é um protótipo? Por qual motivo é usado?

Um protótipo é a declaração de uma função que apresenta somente o nome, tipos e número dos argumentos e seu tipo de retorno (int, float, char, void, etc). O protótipo permite ao compilador que retorne erros caso a função seja chamada incorretamente, seja por falta ou até excesso de parâmetros dentre outros possíveis erros.

* + **Pergunta 2:** O que significa cada um dos dígitos 0666?

O dígito (0) significa a falta de permissão e cada um dos restantes (6 6 6) representam a permissão de leitura e escrita da fila de mensagens criada.

* + **Pergunta 3:** Para que serve o arquivo stderr?

O arquivo stderr (ou standard error stream) serve para registrar as mensagens sobre os erros que ocorreram durante o tempo de execução.

* + **Pergunta 4:** Caso seja executada a chamada fprintf com o handler stderr, onde aparecerá o seu resultado?

O resultado aparecerá na tela do terminal em que foi executado o programa.

* + **Pergunta 5:** Onde stderr foi declarado?

stderr está contido na biblioteca stdio.h, que foi incluída no início do código.

* + **Pergunta 6:** Explicar o que são e para que servem stdin e stdout.

stdin é o fluxo de entrada de dados padrão que permite a entrada de textos. Já stdout são fluxos de saída padrão de textos do comando para o shell.

* + **Pergunta 7:** O que ocorre com a fila de mensagens, se ela não é removida e osprocessos terminam?

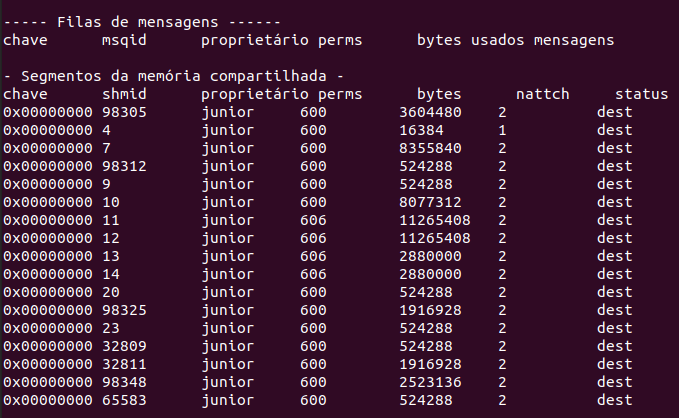
A fila de mensagens é removida da memória.

* + **Pergunta 8:** Qual será o conteúdo de data\_ptr?

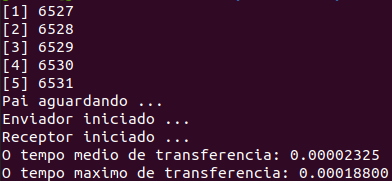
Ao olhar o tipo de data\_ptr percebe-se que a mesma é uma instância do tipo struct data\_t que contém o número da mensagem (msg\_no) e o tempo que será enviada para realizar os cálculos de desvio.

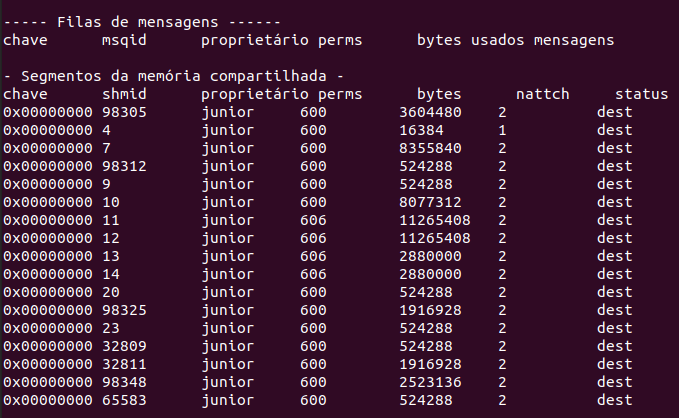
**Resultados da Execução do Programa Exemplo (Parte 1):**

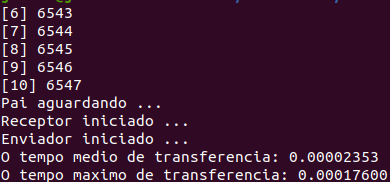
* **Dados recolhidos**
  + **1ª Execução:**
  + **Ipcs:**

****

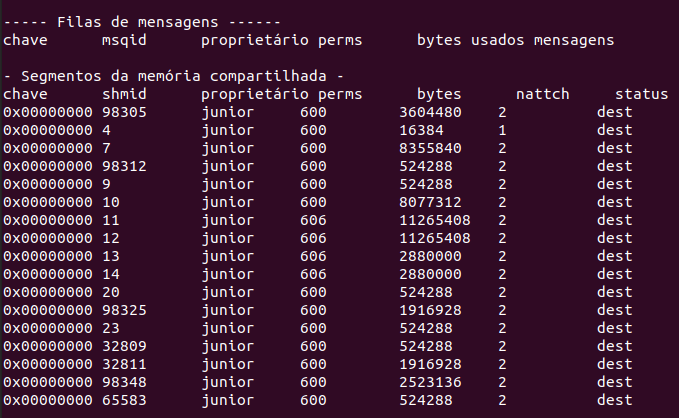
* + **Pids**

****

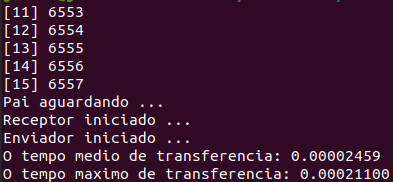
* + **2ª Execução:**
  + **Ipcs:**
  + 
  + **Pids:**

****

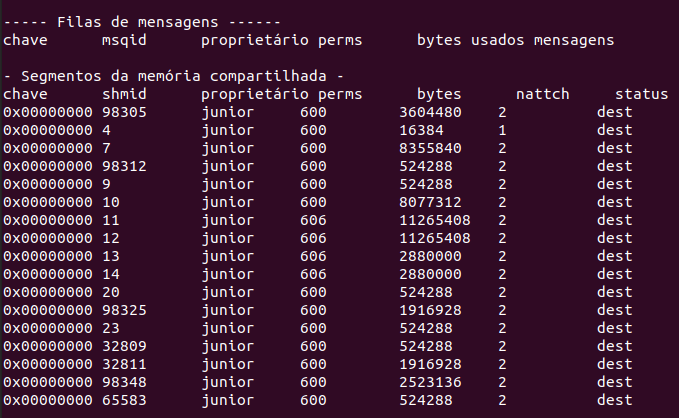
* + **3ª Execução:**
  + **Ipcs:**

****

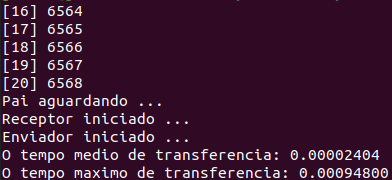
* + **Pids:**

****

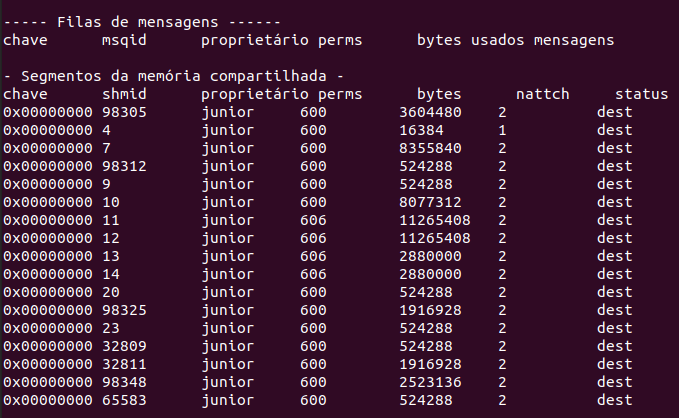
* + **4ª Execução:**
  + **Ipcs:**

****

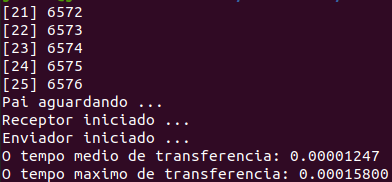
* + **Pids:**

****

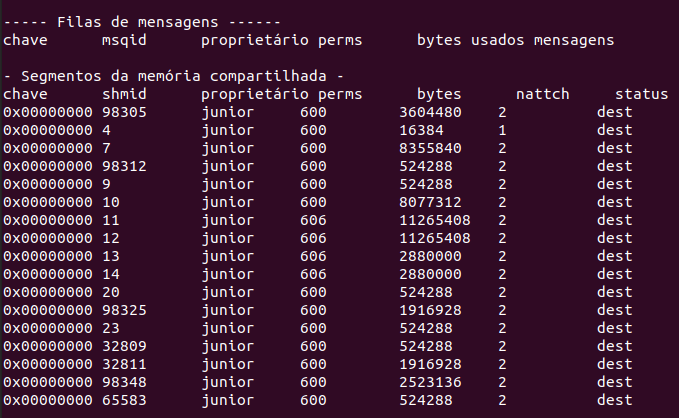
* + **5ª Execução:**
  + **Ipcs:**

****

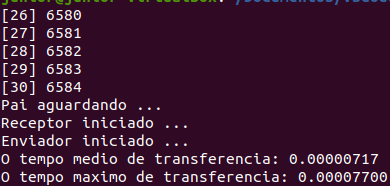
* + **Pids:**

****

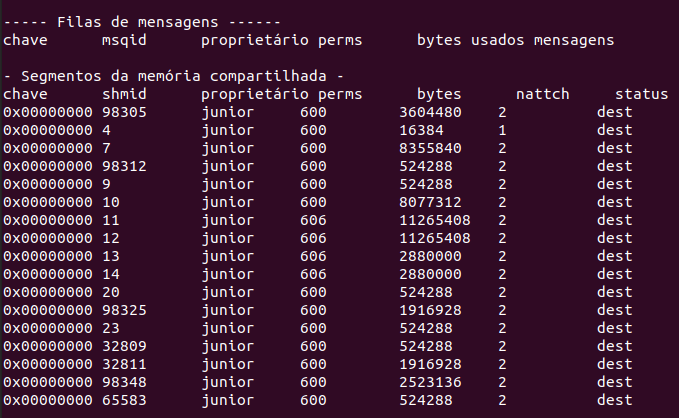
* + **6ª Execução:**
  + **Ipcs:**

****

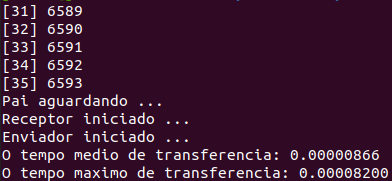
* + **Pids:**

****

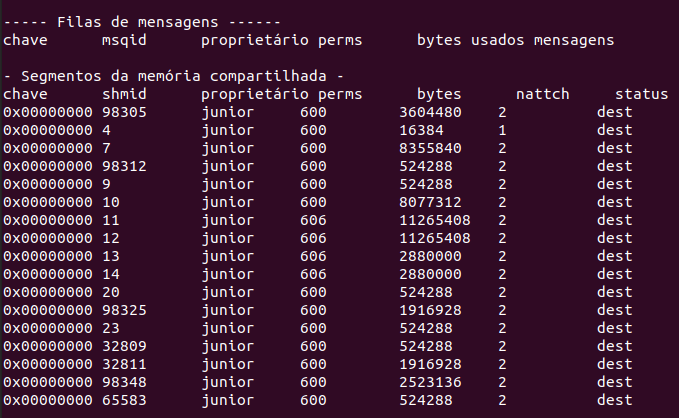
* + **7ª Execução:**
  + **Ipcs:**

****

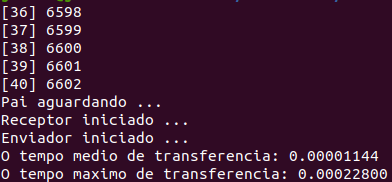
* + **Pids:**

****

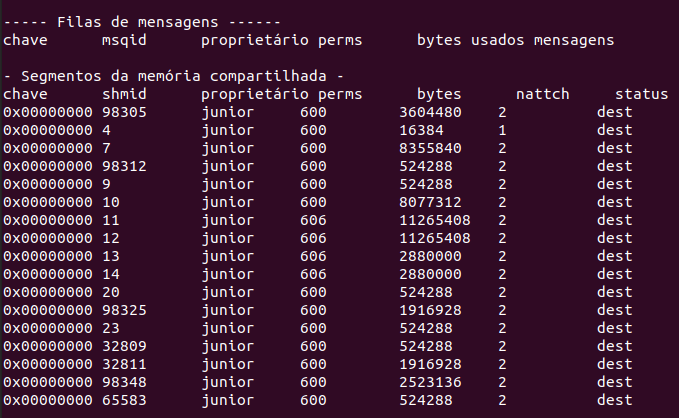
* + **8ª Execução:**
  + **Ipcs:**

****

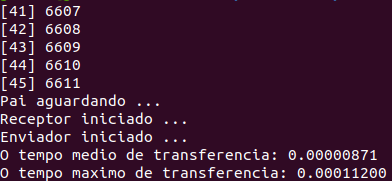
* + **Pids:**

****

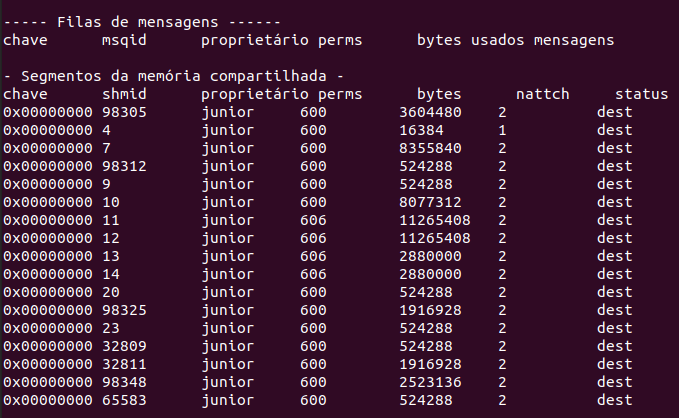
* + **9ª Execução:**
  + **Ipcs:**

****

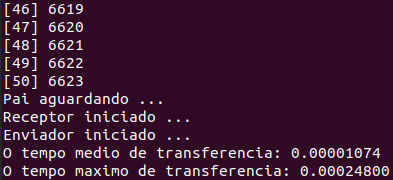
* + **Pids:**

****

* + **10ª Execução:**
  + **Ipcs:**

****

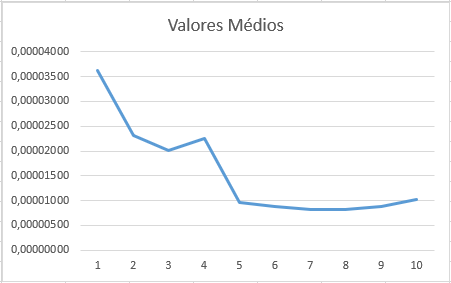
* + **Pids:**

****

* Ao observar as chamadas IPCS, nota-se que não há filas de mensagens após a execução do código, comprovando que são removidas do sistema operacional, além disso, a estruturação dos segmentos de memória compartilhada se mantém iguais ao decorrer das 10 execuções.
* **Tabela Gerada**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Execução | Médio(seg) | Máximo(seg) | Carga | Console |
| 1 | 0,00003618 | 0,00032000 | 5 | Mesmo |
| 2 | 0,00002312 | 0,00024000 | 10 | Mesmo |
| 3 | 0,00002012 | 0,00018400 | 15 | Mesmo |
| 4 | 0,00002258 | 0,00018000 | 20 | Mesmo |
| 5 | 0,00000953 | 0,00014000 | 25 | Mesmo |
| 6 | 0,00000875 | 0,00018200 | 30 | Mesmo |
| 7 | 0,00000817 | 0,00014300 | 35 | Mesmo |
| 8 | 0,00000822 | 0,00010700 | 40 | Mesmo |
| 9 | 0,00000880 | 0,00017400 | 45 | Mesmo |
| 10 | 0,00001020 | 0,00022300 | 50 | Mesmo |

* **Gráficos**

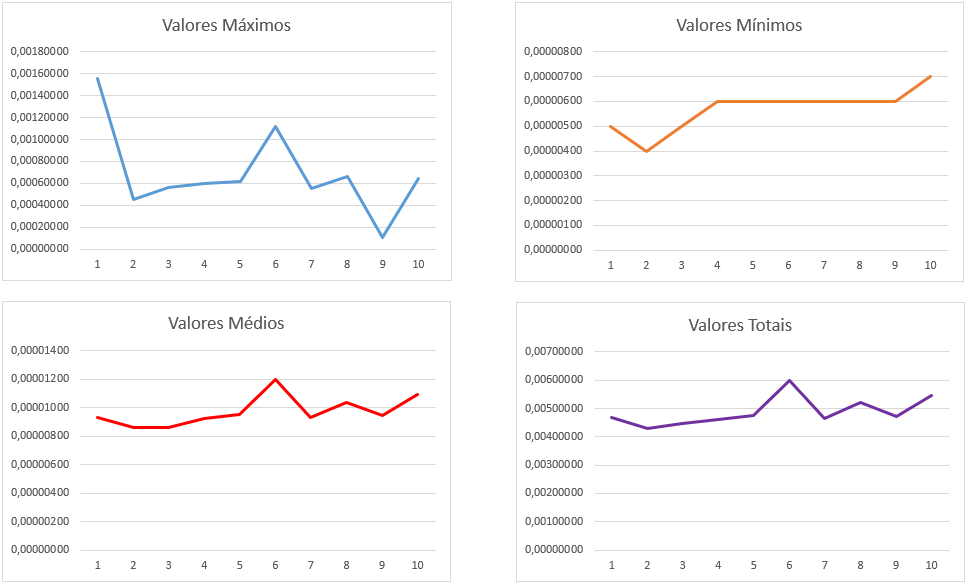
****



**Resultados da Execução do Arquivo Modificado (Parte 2):**

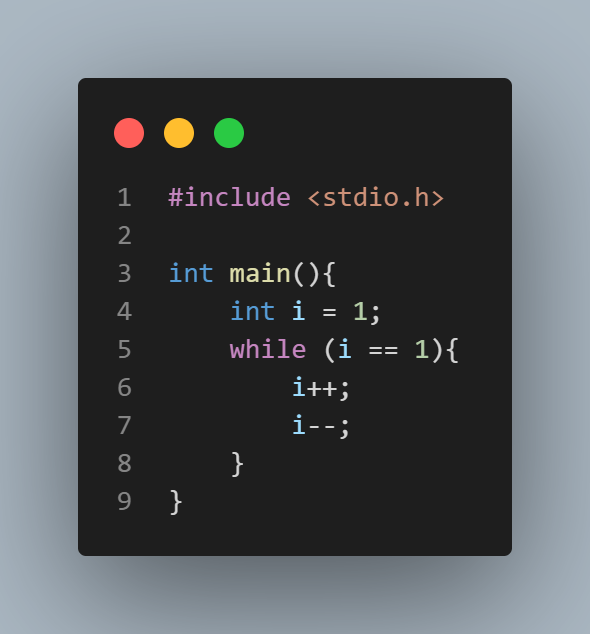
* **Dados Recolhidos**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Execução | Máximo(seg) | Mínimo(seg) | Médio(seg) | Total(seg) | Tamanho da Mensagem |
|  |
| 1 | 0,00156000 | 0,00000500 | 0,00000934 | 0,00467102 | 512 |  |
| 2 | 0,00044900 | 0,00000400 | 0,00000862 | 0,00431200 | 1024 |  |
| 3 | 0,00056601 | 0,00000500 | 0,00000865 | 0,00447502 | 1536 |  |
| 4 | 0,00060100 | 0,00000600 | 0,00000926 | 0,00462800 | 2048 |  |
| 5 | 0,00061500 | 0,00000600 | 0,00000953 | 0,00476500 | 2560 |  |
| 6 | 0,00111800 | 0,00000600 | 0,00001196 | 0,00597798 | 3072 |  |
| 7 | 0,00055500 | 0,00000600 | 0,00000929 | 0,00464300 | 3584 |  |
| 8 | 0,00066600 | 0,00000600 | 0,00001039 | 0,00519701 | 4096 |  |
| 9 | 0,00010500 | 0,00000600 | 0,00000946 | 0,00473200 | 4608 |  |
| 10 | 0,00064800 | 0,00000700 | 0,00001093 | 0,00546301 | 5120 |  |

* **Gráfico**

**Análise dos resultados:**

* **Programa Exemplo:**
  + **Observando os valores Médios:** ao decorrer das 10 execuções os intervalos de tempo apresentam uma tendência de redução conforme o aumento de carga (5 em 5), porém em certos momentos há um aumento no desvio (por exemplo na quarta execução).
  + **Observando os valores Máximos:** assim como o gráfico dos valores médios, a linha apresenta uma tendência a redução do tempo de transferência com certos picos (por exemplo na sexta execução), porém, a partir da oitava execução a linha assume um comportamento de aumento nos intervalos de transferência dos dados.
* **Programa Modificado:**
  + **Observando os valores Máximos:** a reta apresenta um grande déficit da primeira a segunda execução, a partir daí, por maior q se torne a mensagem, tende a manter um valor médio com pico considerável na sexta execução e outro déficit na nona.
  + **Observando os valores Mínimos:** a reta mantém o comportamento de crescimento leve, sendo constante durante 5 execuções (quarta à nona) e apresentando decréscimo apenas entre a primeira e segunda execuções.
  + **Observando os valores Médios:** comportamento similar aos valores mínimos, com crescimento leve ao decorrer do aumento do tamanho da mensagem, porém assim como os valores máximos, este apresenta um pico na 6 sexta execução do programa.
  + **Observando os valores Totais:** gráfico muito semelhante ao de valores médios, porém com sua ordem de grandeza maior.

**Programa Usado para Aumento de Carga:**

**Conclusão:**

Na primeira etapa, conclui-se que apesar do aumento de cargas o tempo de transferência médio se manteve decrescente, afirmando a eficiência do Multi-Tasking da CPU. E em relação à segunda etapa, apesar do aumento do intervalo de tempo não apresentar a constância vista na primeira parte, os mesmos não foram tão impactados com o aumento da mensagem.