

Experimento #2

Sistemas Operacionais A



Adriano de Oliveira Munin	RA:17066960
Fábio Seiji Irokawa	RA:17057720
Lucas Rodrigues Coutinho	RA:17776501
Marcos Lelis de F. Oliveira	RA:16248387
Paulo M. Biocchi	RA:16148363

Introdução

Neste projeto, realizamos dois experimentos, nos quais consistiam em corrigir erros e executar um programa (experimento 1), tal programa criava dois filhos na main(), sendo que um dos filhos envia uma mensagem por IPC contendo dados de hora para o outro filho, e o outro ao receber tal mensagem pega outros dados de hora atualizados e calcula a diferença, a seguir, imprime os dados na tela. Já o segundo experimento consiste em modificar o primeiro programa para que a impressão de dados na tela seja feita por um terceiro filho, e para isso criou-se uma segunda fila de mensagens para transmitir os dados de tempo já computados para a impressão.

Todo o projeto tem como objetivo o aprendizado sobre como funciona e a importância da comunicação entre os processos através da fila de mensagens. É necessário saber o funcionamento dessa troca de informações entre os processos, já que são filhos do mesmo pai e precisam dessa comunicação para a realização do programa em execução. Para os testes realizados nesse experimento foi utilizado um processador Core i7 com 8 MB de cache, frequência de 4GHz e 4 núcleos físicos.

Resultados da Execução

Primeira etapa do experimento

Foram realizados testes com o número de interações fixado em 500 e foi executada uma carga no mesmo console cuja a quantidade foi aumentada de 5 em 5 durante as 10 execuções.

Tabela 1- tempos médio e máximo de transferência de mensagens

Execução	Médio(seg.)	Máximo(seg.)	Carga	No mesmo console
1	0.000007	0.000041	0	Sim
2	0.000138	0.011880	5	Sim
3	0.000458	0.013647	10	Sim
4	0.000155	0.011597	15	Sim
5	0.000603	0.017425	20	Sim
6	0.000216	0.016738	25	Sim
7	0.000385	0.017608	30	Sim
8	0.000617	0.016013	35	Sim
9	0.000326	0.011944	40	Sim
10	0.000259	0.013296	45	Sim

A imagem abaixo mostra um exemplo de execução da primeira etapa do experimento, nela é possível ver a criação de um processo de carga com 15 filhos que executam uma rotina de loop infinito, esses 15 processo irão executar concorrentemente com o programa do experimento que é executado logo em seguida no mesmo terminal

```

lucas@lucas-MS-7821: ~/Downloads/Experimento2
Arquivo Editar Ver Pesquisar Terminal Ajuda
lucas@lucas-MS-7821:~$ cd Downloads/Experimento2/
lucas@lucas-MS-7821:~/Downloads/Experimento2$ ./carga
Digite a quantidade de cargas
15
^Z
[1]+  Parado                  ./carga
lucas@lucas-MS-7821:~/Downloads/Experimento2$ bg
[1]+ ./carga &
lucas@lucas-MS-7821:~/Downloads/Experimento2$ ./experimento2
Pai aguardando ...
Emissor iniciado ...
Receptor iniciado ...
0 tempo medio de transferencia: 0.000004
0 tempo maximo de transferencia: 0.000679

```

Figura 1- Processo de carga e experimento sendo executados

Aqui vemos o resultado da execução do programa de carga junto com o experimento, na imagem é possível ver o uso das CPUs em 100%.

```

lucas@lucas-MS-7821: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda

 1  [|||||||||||||||||||||||||||||||||100.0%]  5  [|||||||||||||||||||||||||||||||||100.0%]
 2  [|||||||||||||||||||||||||||||||||100.0%]  6  [|||||||||||||||||||||||||||||||||100.0%]
 3  [|||||||||||||||||||||||||||||||||100.0%]  7  [|||||||||||||||||||||||||||||||||100.0%]
 4  [|||||||||||||||||||||||||||||||||100.0%]  8  [|||||||||||||||||||||||||||||||||100.0%]
Mem[|||||||||||||||||||||||||||||4.41G/7.73G]  Tasks: 211, 993 thr; 8 running
Swp[|||||||||||||||||||||||||0K/2.00G]      Load average: 16.85 25.16 25.70
                                           Uptime: 04:25:52

  PID USER      PRI  NI  VIRT   RES   SHR  S  CPU% MEM%   TIME+  Command
  ---  ---      -
1340 gdm        20   0 111M  2812   32 S   0.0  0.0  0:00.00 (sd-pam)
 9203 lucas      20   0 255M  2852   32 S   0.0  0.0  0:00.00 (sd-pam)
29570 lucas     20   0 4508    80    0 R  47.2  0.0  1:42.50 ./carga
29579 lucas     20   0 4508    80    0 R  45.9  0.0  1:39.70 ./carga
29576 lucas     20   0 4508    80    0 R  47.2  0.0  1:41.00 ./carga
29565 lucas     20   0 4508   820   756 R  45.9  0.0  1:40.63 ./carga
29575 lucas     20   0 4508    80    0 R  49.1  0.0  1:40.69 ./carga
29568 lucas     20   0 4508    80    0 R  49.8  0.0  1:40.23 ./carga
29577 lucas     20   0 4508    80    0 R  47.2  0.0  1:42.95 ./carga
29580 lucas     20   0 4508    80    0 R  43.9  0.0  1:40.22 ./carga
29573 lucas     20   0 4508    80    0 R  46.5  0.0  1:41.08 ./carga
29572 lucas     20   0 4508    80    0 R  49.8  0.0  1:40.14 ./carga
29578 lucas     20   0 4508    80    0 R  43.2  0.0  1:40.26 ./carga
29571 lucas     20   0 4508    80    0 R  48.5  0.0  1:40.63 ./carga
29569 lucas     20   0 4508    80    0 R  49.1  0.0  1:41.49 ./carga
29574 lucas     20   0 4508    80    0 R  43.2  0.0  1:40.09 ./carga
29566 lucas     20   0 4508    80    0 R  42.6  0.0  1:38.67 ./carga
29567 lucas     20   0 4508    80    0 R  47.8  0.0  1:41.13 ./carga
29752 lucas     20   0 4512   756   692 S   0.0  0.0  0:00.00 ./experimento2
18962 lucas     20   0 1781M 33592 11920 S   0.0  0.4  0:00.45 /home/lucas/.vscode/extensions/ms-
18963 lucas     20   0 1781M 33592 11920 S   0.0  0.4  0:03.21 /home/lucas/.vscode/extensions/ms-
18953 lucas     20   0 1781M 33592 11920 S   0.0  0.4  0:07.67 /home/lucas/.vscode/extensions/ms-
18968 lucas     20   0 1781M 33592 11920 S   0.0  0.4  0:02.34 /home/lucas/.vscode/extensions/ms-

```

Figura 2- PID dos processos de carga e experimento

Aqui vemos três tipos de comunicação entre processos, filas de mensagens, segmentos de memória e arrays de semáforos. Para este experimento iremos utilizar filas de mensagens. Na primeira parte do experimento foi criada uma fila com a chave 3102 que em hexa é 0x00000C1E, a partir dessa chave é possível obter o msqid da fila que nessa execução tem o valor de 20387456.

```
lucas@lucas-MS-7821: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
0x00000000 1015816 lucas 600 524288 2 dest
0x00000000 1245194 lucas 600 16777216 2 dest

----- Arrays de semáforos -----
chave semid proprietário perms nsems

lucas@lucas-MS-7821:~$ ipcs

----- Filas de mensagens -----
chave msqid proprietário perms bytes usados mensagens
0x00000c1e 10387456 lucas 666 0 0

- Segmentos da memória compartilhada -
chave shmid proprietário perms bytes nattch status
0x00000000 262144 lucas 600 134217728 2 dest
0x00000000 360449 lucas 600 524288 2 dest
0x00000000 393218 lucas 600 524288 2 dest
0x00000000 425987 lucas 600 524288 2 dest
0x00000000 2818052 lucas 600 524288 2 dest
0x00000000 786437 lucas 600 524288 2 dest
0x00000000 819206 lucas 600 524288 2 dest
0x00000000 917511 lucas 600 524288 2 dest
0x00000000 1015816 lucas 600 524288 2 dest
0x00000000 1245194 lucas 600 16777216 2 dest

----- Arrays de semáforos -----
chave semid proprietário perms nsems

lucas@lucas-MS-7821:~$
```

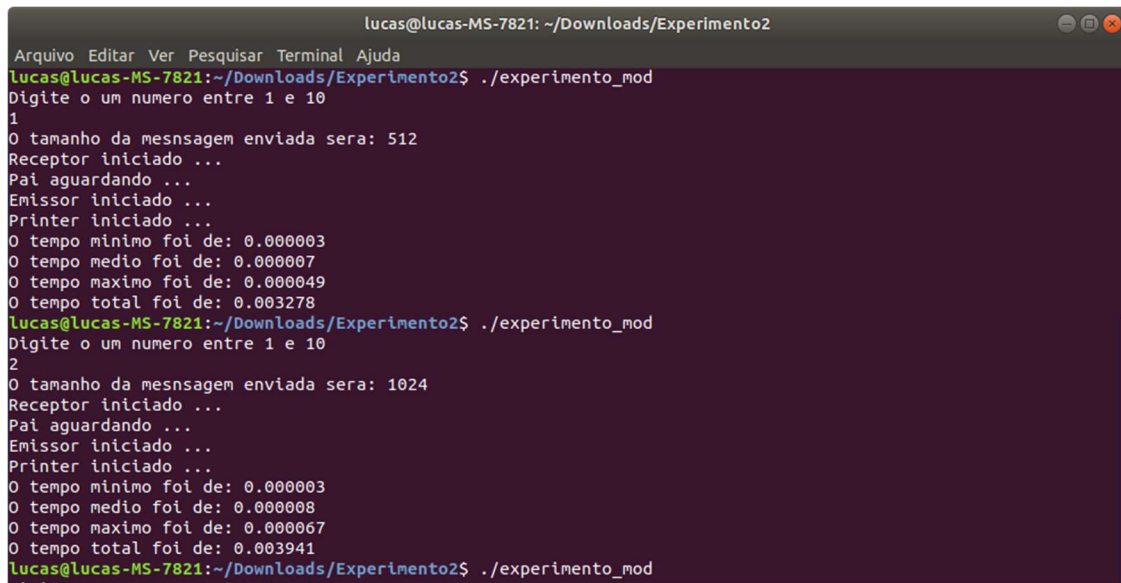
Figura 3- Detalhes dos modos de transferência de mensagens

Segunda etapa do experimento

Foram realizados testes variando o tamanho da mensagem transmitida, com isso foram obtidos os tempos de transmissão da mensagem mostrados na tabela abaixo.

Execução	Mínimo(seg.)	Médio(seg.)	Máximo(seg.)	Total(seg.)	tamanho da mensagem
1	0.000003	0.000007	0.000049	0.003278	512
2	0.000003	0.000008	0.000067	0.003941	1024
3	0.000001	0.000006	0.000302	0.003062	1536
4	0.000003	0.000008	0.000048	0.003867	2048
5	0.000002	0.000006	0.000145	0.003205	2560
6	0.000002	0.000008	0.000135	0.003807	3072
7	0.000003	0.000009	0.000154	0.004602	3584
8	0.000002	0.000009	0.000110	0.004305	4096
9	0.000004	0.000009	0.000132	0.004322	4608
10	0.000003	0.000009	0.000067	0.004449	5120

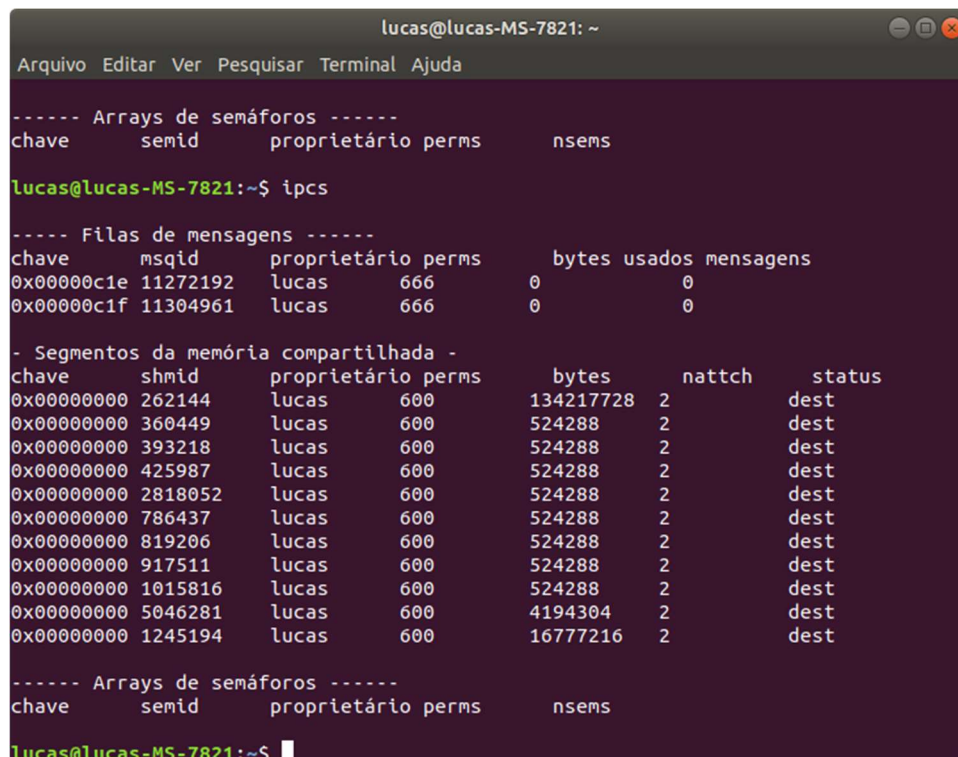
Na imagem abaixo é possível ver parte do teste variando o tamanho da mensagem a ser enviada, mantendo o processador sem carga.



```
lucas@lucas-MS-7821: ~/Downloads/Experimento2
Arquivo Editar Ver Pesquisar Terminal Ajuda
lucas@lucas-MS-7821:~/Downloads/Experimento2$ ./experimento_mod
Digite o um numero entre 1 e 10
1
O tamanho da mensagem enviada sera: 512
Receptor iniciado ...
Pai aguardando ...
Emissor iniciado ...
Printer iniciado ...
O tempo minimo foi de: 0.000003
O tempo medio foi de: 0.000007
O tempo maximo foi de: 0.000049
O tempo total foi de: 0.003278
lucas@lucas-MS-7821:~/Downloads/Experimento2$ ./experimento_mod
Digite o um numero entre 1 e 10
2
O tamanho da mensagem enviada sera: 1024
Receptor iniciado ...
Pai aguardando ...
Emissor iniciado ...
Printer iniciado ...
O tempo minimo foi de: 0.000003
O tempo medio foi de: 0.000008
O tempo maximo foi de: 0.000067
O tempo total foi de: 0.003941
lucas@lucas-MS-7821:~/Downloads/Experimento2$ ./experimento_mod
```

Figura 4- Execução do experimento variando tamanho das mensagens

Aqui é possível visualizar as duas filas que foram criadas para transmissão das mensagens, sendo que uma delas foi utilizada para transmitir a hora atual para o processo que calcula o tempo e a outra para transmitir os tempos calculados para o processo que imprime.



```
lucas@lucas-MS-7821: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda

----- Arrays de semáforos -----
chave      semid      proprietário perms      nsems

lucas@lucas-MS-7821:~$ ipcs

----- Filas de mensagens -----
chave      msqid      proprietário perms      bytes usados mensagens
0x00000c1e 11272192   lucas      666      0      0
0x00000c1f 11304961   lucas      666      0      0

- Segmentos da memória compartilhada -
chave      shmid      proprietário perms      bytes      nattch      status
0x00000000 262144     lucas      600      134217728  2      dest
0x00000000 360449     lucas      600      524288     2      dest
0x00000000 393218     lucas      600      524288     2      dest
0x00000000 425987     lucas      600      524288     2      dest
0x00000000 2818052    lucas      600      524288     2      dest
0x00000000 786437     lucas      600      524288     2      dest
0x00000000 819206     lucas      600      524288     2      dest
0x00000000 917511     lucas      600      524288     2      dest
0x00000000 1015816    lucas      600      524288     2      dest
0x00000000 5046281    lucas      600      4194304    2      dest
0x00000000 1245194    lucas      600      16777216   2      dest

----- Arrays de semáforos -----
chave      semid      proprietário perms      nsems

lucas@lucas-MS-7821:~$
```

Figura 5- Duas filas criadas

Análise dos Resultados

Primeiro programa com variações de carga

Foram obtidas duas tabelas, a primeira é em relação ao programa exemplo original, no qual foi aferido os tempos gastos em 10 execuções. Em cada execução, foram acrescentadas 5 cargas que ocupam o processador, como se pode observar na primeira tabela. E a segunda, refere-se ao programa modificado.

Verifica-se que quanto mais carregado o processador estiver, mais lento será o tempo de troca das mensagens (Tempo máximo).

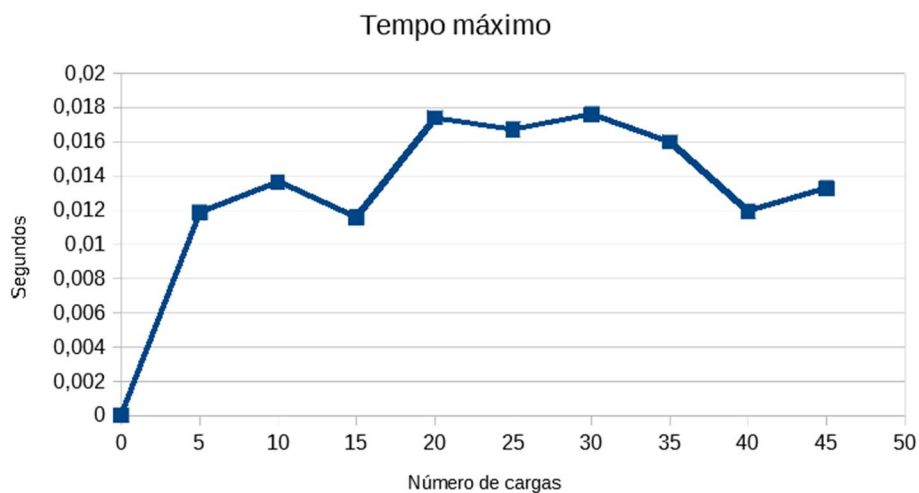


Figura 6- Gráfico com variação do tempo máximo de transferência das mensagens

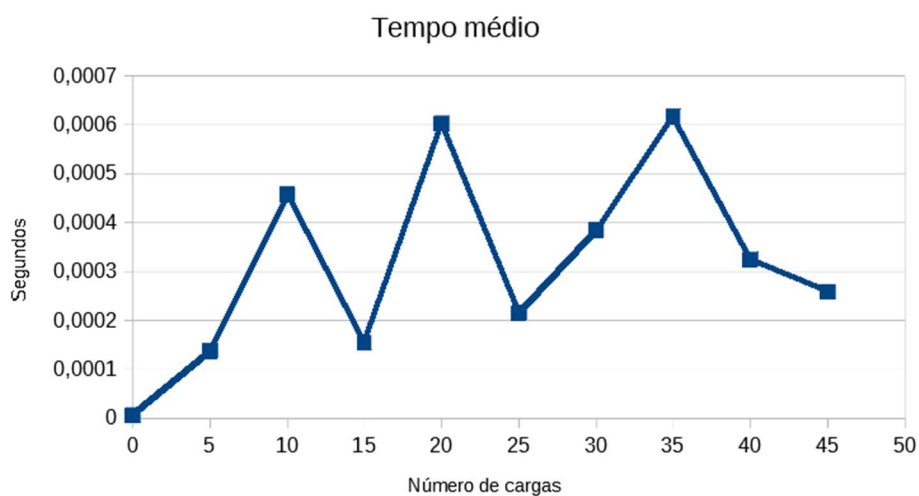


Figura 7- Gráfico com variação do tempo médio de transferência das mensagens

Segundo Programa com variações no tamanho das mensagens

Na segunda tabela, temos os dados do programa modificado, que a cada execução o tamanho da mensagem trocada aumenta, portanto quanto maior a mensagem, maior poderá ser o tempo gasto para ela trafegar.

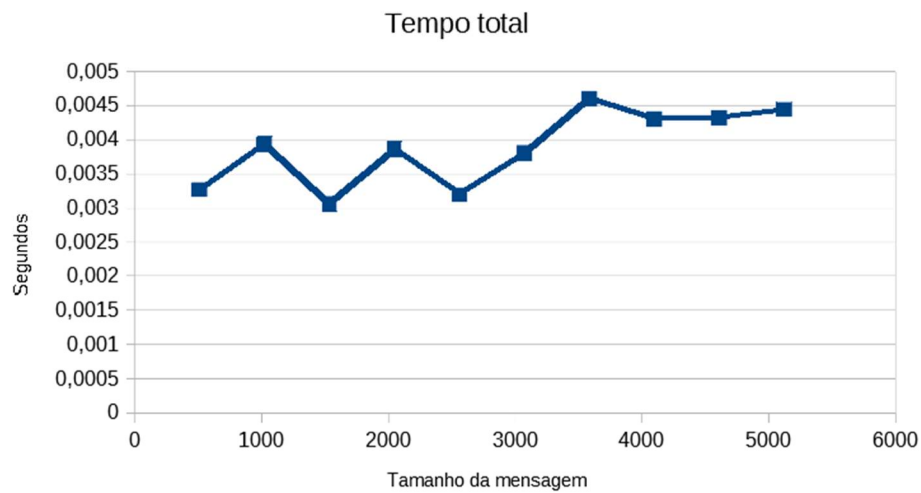


Figura 8- Gráfico com tempo total de transferência das mensagens

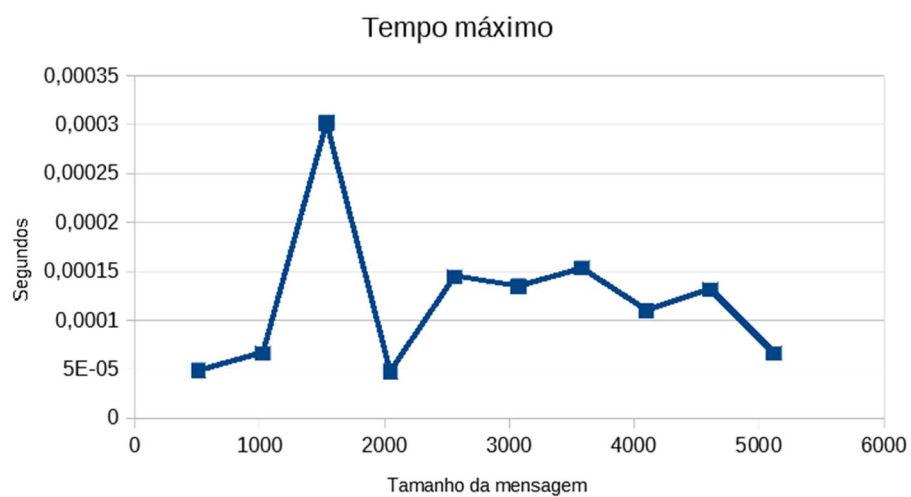


Figura 9- Gráfico com tempo médio de transferência das mensagens

Erros no Código

Original → Modificado

Linha 75:

#include <types.h> → #include <sys/types.h>

Linha 77:

#include <ipc.h> → #include <sys/ipc.h>

Linha 78:

#include <msg.h> → #include <sys/msg.h>

Faltava:

#include <stdlib.h>

Faltava:

#define NO_OF_CHILDREN 2

Linha 122:

int rtn; → int rtn = 1;

Linha 123:

int count = 10; → int count = 0;

Linha 154:

for(count = 0; count < NO_OF_CHILDREN; count-) → for(count = 0; count < NO_OF_CHILDREN; count++)

Linha 156:

rtn == fork; → rtn = fork();

Linha 158:

exit(NULL); → break;

Linha 196:

if(msgctl(queue_id,IPC_RMID,NULL) == 0) → if(msgctl(queue_id,IPC_RMID,NULL) != 0)

Linha 263:

for(count = 0; count < NO_OF_ITERATIONS; ++count) → for(count = 0; count < NO_OF_ITERATIONS; count++)

Linha 280:

delta -= receive_time.tv_sec - data_ptr->send_time.tv_sec; → delta = receive_time.tv_sec - data_ptr->send_time.tv_sec;

Linha 281:

delta = (receive_time.tv_usec - data_ptr->send_time.tv_usec)/(float)MICRO_PER_SECOND;
→ delta=(receive_time.tv_usec-data_ptr->send_time.tv_usec)/(float)MICRO_PER_SECOND;

Linha 282:

total +=- delta; → total += delta;

Linha 287:

if(delta < max) → if(delta > max)

Perguntas do código

- 1: O protótipo de uma função é basicamente a pré-declaração da função em si. Ele é considerado uma boa prática de programação porque organiza o código e o deixa melhor visualmente.
- 2: Os números de cada dígito estão em octal, o qual transformando 6 octal em binário representa 110, onde cada número é um tipo de permissão sendo respectivamente leitura, escrita e execução com 1 sendo permitido e 0 proibido. E cada posição dos 6 representa permissão para o dono, grupo e outro respectivamente.
- 3: É um destinatário padrão para mensagens de erro e outros diagnósticos de avisos.
- 4: O erro aparecerá no console do Linux
- 5: É declarado na biblioteca `stdint.h`
- 6: `Stdin` ou `Standard input` é o arquivo handle que o processo lê para obter a informação do usuário. `Stdout` ou `standard output` é a escrita da informação no arquivo handle.
- 7: A fila de mensagens não pode mais ser removida, somente por meio da chamada `ipcrm`.

Perguntas do documento PDF

- 1: Berkeley Unix é uma ramificação do Unix da AT&T. System V é uma das primeiras versões de sistema operacional Unix. Posix ou Interface Portável entre Sistemas Operacionais é uma família de normas definidas pelo IEEE (Instituto de Engenheiros Eletricistas e Eletrônicos) para a manutenção de compatibilidade entre sistemas operacionais. AT&T ou American Telephone and Telegraph é uma companhia americana de telecomunicações. Socket permite a comunicação entre dois processos diferentes tanto na mesma máquina quanto em máquinas de cliente e servidor. Fila de mensagens é a comunicação entre processos ou threads de um mesmo processo. Memória compartilhada é um espaço na memória onde os processos têm acesso para compartilhar as informações entre eles. Pipe permite a comunicação de um sentido de um processo ao outro
- 2: As chamadas `ipc` e `ipcrm` apresentam informações sobre os recursos de IPC (Inter Process Communication), sendo a chamada `ipcs` para listar os recursos utilizados, como semáforos e filas de mensagens e a chamada `ipcrm` para encerrar os recursos de IPC utilizados pelo programa.
- 3: O handle devolvido pela chamada `msgget` é o id da fila criada, este id é utilizado tanto para enviar mensagens para a fila quanto para receber da mesma, inclusive este id também é utilizado para encerrar a fila criada. Já a chave única (key) é utilizada para criar uma fila ou para obter o id de uma fila já criada.
- 4: Sim. Como padrão é definido que o tamanho máximo de uma mensagem é de: 8192 bytes.
- 5: Sim. Como padrão é definido que o tamanho máximo de uma fila de mensagens é de: 16384 bytes.
- 6: `Typedef` permite a definição de um nome para um determinado tipo.
- 7: `Typedef` deve ser usado na criação de uma struct, que irá definir o nome dessa nova struct.

Conclusão

Na primeira parte do experimento, antes de serem realizados os testes com carga no processador, tanto o tempo médio e máximo de transferência estavam na casa dos microssegundos. Após ser iniciada a primeira rodada de cargas que executou 5 processos, os tempos médio e máximo foram para a casa dos milissegundos, as demais variações de cargas feitas a partir das 5 iniciais, não afetaram de forma significativa os tempos. Com isso podemos concluir que quando todos os núcleos estão sobrecarregados, o tempo de transmissão das mensagens aumenta, independentemente da quantidade de cargas.

Na segunda parte, houve uma oscilação, depois um leve aumento nos tempos conforme o tamanho da mensagem foi aumentando, porém em alguns casos, mesmo com o tamanho das mensagens sendo grande o tempo fica abaixo da média. Nesse caso como as mensagens são passadas por ponteiros para regiões de memória, o tamanho da mensagem possa não ter tanta influência nos tempos.