MAC0422 – Sistemas Operacionais

EP1- Shell e Simulador de Processos

Marcos Yamazaki - NUSP: 7577622

Caio Lopes - NUSP: 7991187

ep1sh.c

- Neste programa foi implementado um shell bem simples, em que é possível apenas a execução de aluns comandos.

$ cd [diretório]

Este é o primeiro comando a ser verificado, porque não são criado nenhum processo com a execução deste commando.

No código é feita a chamada da função `chdir("diretorio");` que muda o local que está sendo executada o programa.

- Na execução de outros commandos, é feita por Chamadas de Sistema, onde duplicamos o processo, e enquanto o processor “pai” espera pela execução do outro processo, o processo filho executa os comandos.

```
if(fork() == 0) runcmd(input, pasta_atual);
else wait(NULL);
```

- Enquanto o shell aguarda pela finalização dos comandos, agora é feita uma nova chamada de sistema, para a execução dos binários, sejam eles:

/bin/ls -1

/bin/pwd

./ep1 [argumentos]

```
execv(const char *cmd, const char *arg[]);
```

Esta função acima, ela cria um processo filho com o mesmo pid (identificação) e depois se “mata”.

ep1.c – Simulador de processos

- Para cada processo que esta no arquivo trace, **é criado uma struct**, com todas as informações desse processo, como o tempo de chegada, a duraçao, prioridade, o tempo de deadline, e tempo restante para ser executado e entre outros.
- Quando algum processo chega ao sistema para ser executado, ela pode ter **três estados**: Pausado, Executando ou Concluído.
- **Os processos são executados concorrentemete ao programa principal**, que fica contando o tempo e recebendo os processos que vão chegando, assim como decidindo quem é o próximo a ser executado ou pausado.

- Quando o processo está pausado, ela está esperando a liberação do seu semáforo, para conseguir executar.

No momento que ela esta executando, a thread só fica fazendo operações para saber quando tempo já se foi desde o início da execução.

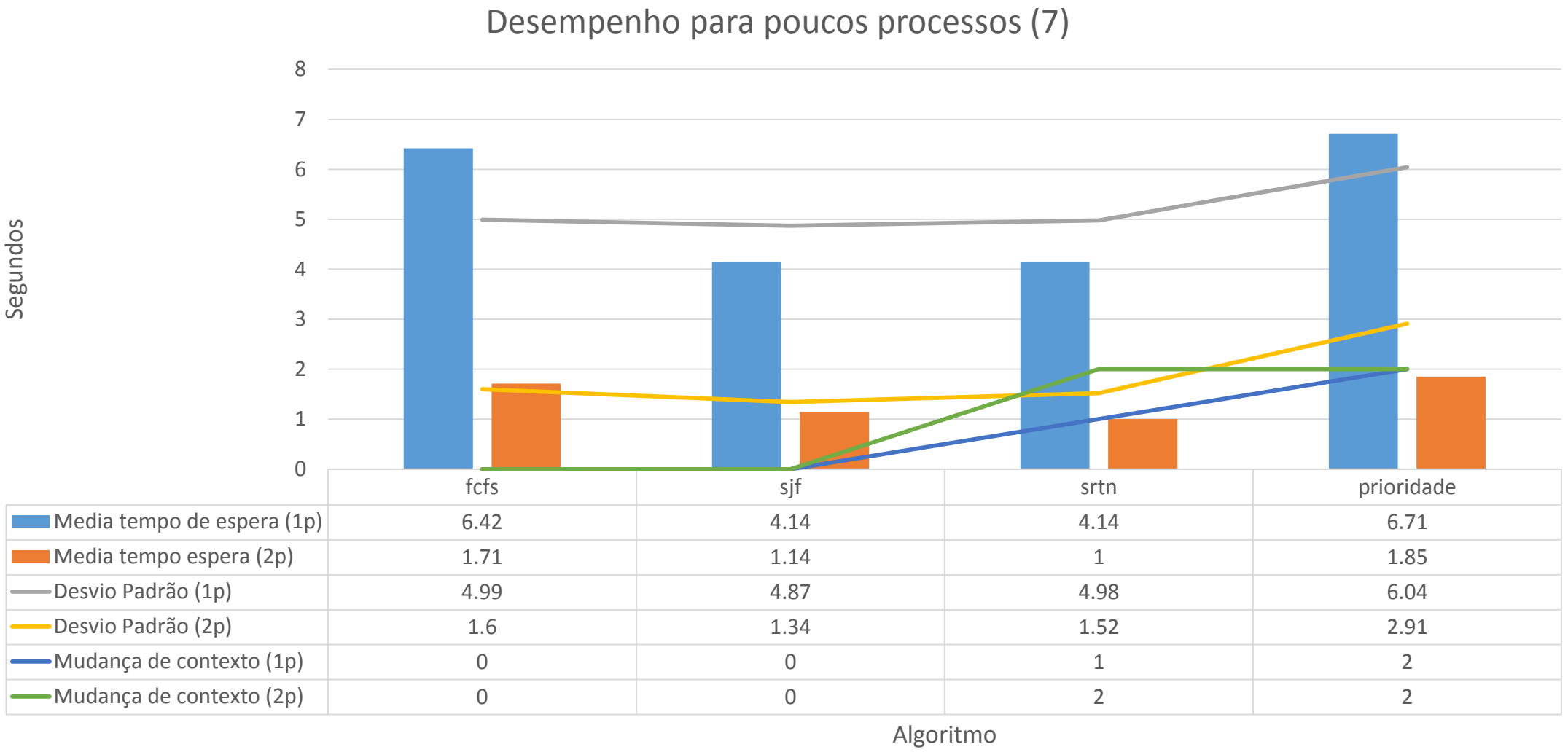
- Para pausar um processo, o escalonador muda o valor da variável estado, da struct do processo que ela quer pausar, assim nessa thread, o processo entra numa parte em que ela terá que esperar a liberação do seu semaforo novamente, que é feita pelo escalonador.

```
sem_wait(&mutex[tinfo->id]);
```

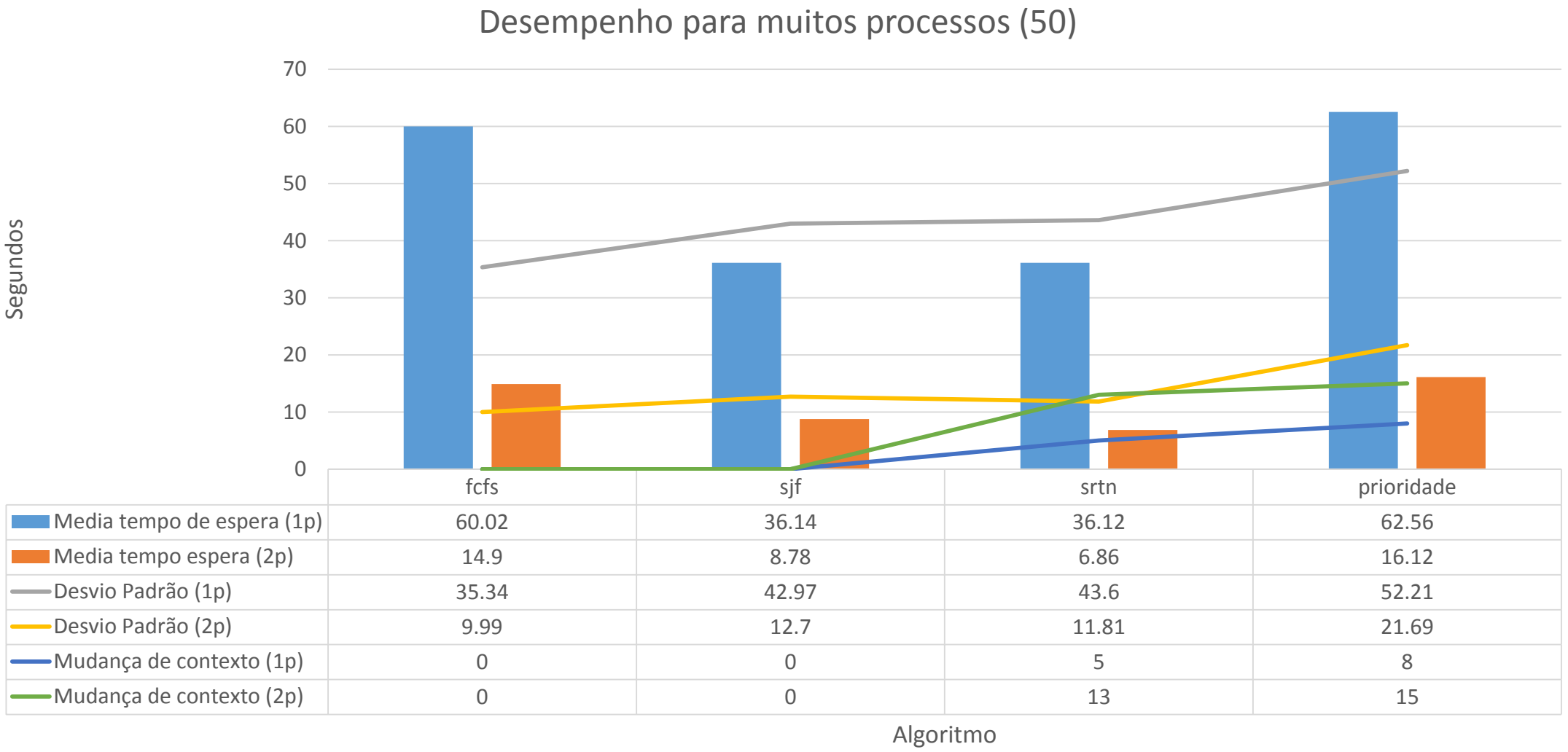
Todos os semáforos são inicializados de modo que não permitam que o processo comece executando assim que ela chegar, só a partir de um sinal, sem_post(&mutex[fila->]); que o processo vai começar.

First-Come First-Served
Shortest Job First
Shortest Remaining Time Next
Escalonamento com prioridade

- Esses quatros escalonadores, foram implementados de modo que quando um processo chega ao sistema para ser executado, é criado uma lista ligada, e colocado cada um na ordem em que elas devem ser rodadas.
- E nos dois últimos casos, quando o processo que acabou de chegar tem um tempo de execução menor que todos os que faltam para serem concluidos, ela pausa o processo de maior tempo faltando, coloca ela para ser o próximo a ser executado caso tenha alguma CPU livre, e começa a rodar esse novo processo.



	fcfs	sjf	srtn	prioridade
Media tempo de espera (1p)	6.42	4.14	4.14	6.71
Media tempo espera (2p)	1.71	1.14	1	1.85
Desvio Padrão (1p)	4.99	4.87	4.98	6.04
Desvio Padrão (2p)	1.6	1.34	1.52	2.91
Mudança de contexto (1p)	0	0	1	2
Mudança de contexto (2p)	0	0	2	2



	fcfs	sjf	srtn	prioridade
Media tempo de espera (1p)	60.02	36.14	36.12	62.56
Media tempo espera (2p)	14.9	8.78	6.86	16.12
Desvio Padrão (1p)	35.34	42.97	43.6	52.21
Desvio Padrão (2p)	9.99	12.7	11.81	21.69
Mudança de contexto (1p)	0	0	5	8
Mudança de contexto (2p)	0	0	13	15

- Pelos gráfico é possivel notar, que no quesito de JUSTIÇA, o escalonamento FCFS, é o que pior nos casos, já que a media do tempo de espera dos processos são bem altas.
- Já nos algoritmos que priorizam o tempo restante, tiveram em media, um tempo de espera media relativamente menor, porém, o desvio padrão desses tempo é mais alta que a própria media.
- Em todos os testes realizados, no quesito de deadline, quando se tinha apenas um processador, a grande maioria dos processos terminaram após seu tempo limite, tendo um melhora significativa ao adicionar um outro processador, e o que tende a melhorar quanto mais núcleos de processamentos a máquina tiver.