Iniciado em	quinta, 13 jan 2022, 20:13
Estado	Finalizada
Concluída em	sábado, 15 jan 2022, 19:40
Tempo empregado	1 dia 23 horas
Avaliar	Ainda não avaliado

Questão 1

Completo

Vale 1,00 ponto(s).

Rode o programa da seguinte forma:

./process-run.py -l 5:100,5:100.

Que cria dois processos com 5 instruções, todas de uso de CPU (100%).

Qual deve ser a utilização da CPU (qual a porcentagem do total turnos que a CPU esteve ocupada)? Por que?

Use as flags -c e -p para rodar novamente e verificar se a sua intuição estava correta

A utilização da CPU seria de 50% para cada processo com as 5 instruções, e o total de turnos que a CPU esteve ocupada é de 100%. Já que ao rodar os processos foi usado 100% do uso da CPU. No entanto, ao "forçar" o uso de 100% da CPU nos dois processos, acabou dividindo de 50% para cada.

Questão 2

Completo

Vale 1,00 ponto(s).

Rode agora:

./process-run -l 4:100,1:0

Que gera um processo de 4 instruções de CPU e um com apenas uma instrução de E/S, que espera o resultado para terminar.

Quanto tempo cada processo vai demorar?

Depois verifique a sua resposta com as flags -c e -p

Tempo de CPU é 5 e tempo do I\O é 4. O primeiro processo usa a CPU, e o segundo processo usa I\O.

Questão 3

Completo

Vale 1,00 ponto(s).

Agora inverta a ordem dos processos da Pergunta 2:

./process-run.py -l 1:0,4:100.

O que vai acontecer agora? Mudar a ordem dos processos vai alterar algo no fim? O que?

(novamente verifique a sua intuição após responder com -c e -p

O tempo da CPU diminui, pois, o I\O não usa a CPU para rodar o processo, e pode ser executada junto com o processo na CPU. Portanto, ao mudar a ordem dos processos diminui o tempo total para 6. Sendo o processo na CPU é 83.33% de ocupação, e I\O de 66.67%.

Questão 4

Completo

Vale 1,00 ponto(s).

A flag -S muda como o sistema reage quando um processo faz E/S. Rode:

./process-run.py -l 1:0,4:100 -c -S SWITCH_ON_END

-S SWITCH_ON_END faz com que o sistema não troque o processo que vai ser bloqueado pra fazer E/S (ou seja, o processo continua na CPU até que chegue a resposta.

O que isto alterou no resultado do simulador?

O tempo total foi agora foi de 9, pois o processo na CPU espera o processo no I\O terminar até voltar a ser executada.

Questão 5

Completo

Vale 1,00 ponto(s).

Agora rode o mesmo processo, mas trocando o comportamento após um processo ser bloqueado para trocar por outro processo, usando -S SWITCH_ON_IO:

./process-run.py -l 1,0,4:100 -c -S SWITCH_ON_IO

O que ocorre agora?

Os processos rodam na CPU e o I\O ao mesmo tempo.

Questão 6

Completo

Vale 1,00 ponto(s).

Outro comportamento importante é o que fazer quando um evento de entrada e saída termina. Com -I IO_RUN_LATER, quando o resultado de E/S chega, o processo que estava esperando por ele volta para a fila de prontos, não sendo necessariamente executado imediatamente. Ao invés disto, o processo que já estava na CPU continua rodando, e ele entra quando chegar a vez dele.

Rode:

./process-run.py 3:0,5:100,5:100,5:100 -S SWITCH_ON_IO -I IO_RUN_LATER -c -p.

O que acontece agora? Os recursos da CPU estão sendo utilizados de forma eficiente?

Não está usando preempção, por isso os processos estão rodando um depois do outro e isso faz com que o tempo aumente.

Questão 7

Completo

Vale 1,00 ponto(s).

Agora repita o processo da pergunta 6, mas usando -I IO_RUN_IMMEDIATE, o que faz com que o processo que recebeu a E/S que chegou seja rodado novamente imediatamente.

Rode:

./process-run.py 3:0,5:100,5:100,5:100 -S SWITCH_ON_IO -I IO_RUN_IMMEDIATE -c -p.

Qual foi a diferença de comportamento? Rodar novamente o processo que fez E/S é uma boa idéia? Por que?

Usando preempção, os processos acabam logo e com menos tempo.

Questão 8

Completo

Vale 1,00 ponto(s).

Agora rode com alguns processos gerados aleatoriamente, como:

-s 1 -l 3:50,3:50
-s 2 -l 3:50,3:50
-s 3 -l 3:50,3:50

o 3:50 significa que o processo terá 3 instruções, e cada uma dela terá 50% de chance de ser de CPU e 50% de E/S (o -s X é para mudar a semente aleatória)

O que acontece quando você roda com esses valores com -I IO_RUN_IMMEDIATE em comparação com -I IO_RUN_LATER?

O que acontece quando você roda com esses valores com -S SWITCH_ON_IO em comparação com -I SWITCH_ON_END?

s 1, 2, 3 - Os processos executam ao mesmo tempo, para -I IO_RUN_IMMEDIATE e -I IO_RUN_LATER

s 1, 2, 3 - No -S SWITCH_ON_IO os processos executam ao mesmo tempo e no -I SWITCH_ON_END espera o processo I\o terminar para poder rodar o processo na CPU.