MOD11-CPU-INIT

TRABALHO PRÁTICO

Neste trabalho prático, executa-se o programa *process-run.py* que permite ver como os estados de um processo muda à medida que os programas correm e utilizam o CPU (isto é, operam uma operação de adição) ou fazem uma E/S (isto é, fazem um pedido para o disco e esperam que termine). Leia o ficheiro *readme-processor_por.txt* para saber utilizar o ficheiro.

1- Corra o programa com as seguintes flags:

python3 process-run.py -l 5:100,5:100

Que utilização deveria ser a do CPU? (e.g. que percentagem de tempo o CPU é utilizado?). Porque razão sabemos a resposta a esta questão?
Acrescente as flags "-c -p" para ver se respondeu certo.

2- Corra o programa agora com as seguintes flags:

python3 process-run.py -l 4:100,1:0

Estas flags dão indicação que um processo tem 4 instruções (todas utilizando CPU) e um que simplesmente faz disparar um evento de E/S e espera até que ele termine. Quanto tempo demorará a completar ambos os processos? Utilize flags "-c -p" para verificar se estava certo.

3- Troque a ordem dos processos executando o seguinte comando:

python3 process-run.py -l 1:0,4:100

O que acontece agora? Será que a troca de ordem interessa? Porquê? (Use "-c -p" para confirmar a sua resposta).

4- Exploremos agora outras possíveis flags. Uma flag importante é -S, que determina como é que o sistema reage quando um processo lança um evento de E/S. Com a flag com o valor SWITCH_ON_END, o sistema não vai trocar com outro processo enquanto decorre o evento E/S, esperando até que o processo termine. O que acontece quando corre os dois processos seguintes, um executando E/S e outro executando trabalho de CPU?

python3 process-run.py -l 1:0,4:100 -c -S SWITCH_ON_END

5- Corre os mesmos processos mas agora com o comportamento de troca de processos (switching) de forma a que um processo é trocado por outro sempre que um outro adquire o estado de WAITING numa E/S:

python3 process-run.py -l 1:0,4:100 -c -S SWITCH ON IO

O que acontece agora? Utilize as flags "-c -p" para confirmar as suas respostas.

6- Um outro comportamento importatnte é o que acontece quando uma E/S termina. Com a flag **-I IO_RUN_LATER**, quando uma E/S termina, o processo que a lançou não reinicia imeiatamente, Em vez disso, o processo que corre nesse momento, mantém-se em execução. O que acontece quando executa a seguinte combinação de processos?

python3 process-run.py -l 3:0,5:100,5:100,5:100 -S SWITCH_ON_IO -I IO_RUN_LATER -c -p

Será que os recursos dos sistema estão a ser devidamente utilizados?

7- Corra agora o mesmo processo mas com a flag ***-I IO_RUN_IMMEDIATE*** que imediatamente inicia a execução do processo que lançou E/S. Como é que este comportamento difere do anterior? Por que razão correr um processo imediatamente a ter terminado o evento E/S é uma boa ideia?

8- Corra agora processos gerados com alguma aleatoriedade, por exemplo , processos com as flags:
-s 1 -l 3:50,3:50, -s 2 -l 3:50,3:50, -s 3 -l 3:50,3:50
Tente prever qual é o resultado da execução destes processos.
O que acontece quando se utiliza o comando ***-I IO_RUN_IMMEDIATE vs. -I IO RUN LATER***? O que acontece quando se utiliza o comando ***-S SWITCH_ON_IO vs. -S SWITCH_ON_END*** ?