

Class Notes Sept 13 2006.

The Clone system call is similar to Fork, except a light-weight process is created that shares all data with the parent process. This represents a kernel thread, because the thread is known to the OS.

There are three types of multithreading.

One-to-One: using only Clone type threads where the OS knows about the threads. These threads can be used to multi-thread with system calls like I/O.

Many-to-One using only something like the pthreads package. You have many user threads, but only one kernel thread, so when a system call occurs, all threads block.

Many-to-Many mixing Clone with pthreads where there are kernel threads and user threads.

Threading issues:

What happens on an exec system call? This replaces the current process with a new process. How does this work if there are multiple kernel threads? Do they all go away? (In Linux, only the thread executing the exec system call is replaced.) What happens with a signal? Do all threads get signaled or only a specific thread? (In Linux, the kernel thread acts as a process with respect to signals. Only one thread gets signaled.)

Scheduling.

Recall how an interrupt is processed.

Interrupt.

Save PC, PSW, and all registers

Figure out what happened

Respond to what happened

Schedule a new Process <-----

Load Registers, including PC & PSW.

That is the step we want to talk about. There are many different algorithms that can be used for scheduling.

Scheduling means choosing a process from the ready queue and running it. Of course, processes make many passes through the ready queue and the wait queues. We are concerned only with one pass through the ready queue, which is called a CPU burst. A pass through the wait queue is called an I/O burst (even if it isn't waiting for I/O).

In most cases we assume that processes will complete their CPU burst when entering the run queue, although this isn't always true.

There are several measures of quality we want to apply to scheduling algorithms. In real operating systems the following are used.

1. CPU Utilization – the amount of time actually spent executing code, as opposed to sitting idle. We will not be concerned with this measure, because it deals with broader issues than picking processes from the ready queue.

2. Throughput – The number of processes completed per unit time. This also is based on broader issues.

3. Wait time – The amount of time a process spends in the ready queue without being executed.

4. Turnaround Time – The amount of time between entering the ready queue and completion of execution.

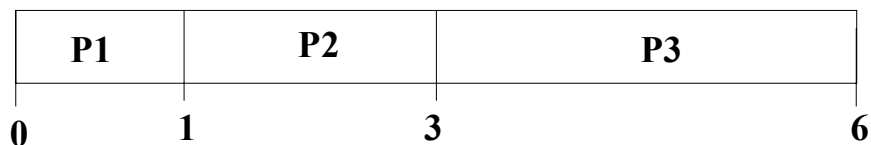
5. Response Time – The amount of time between entering the ready queue and start of execution. The same as wait time for most algorithms.

GANTT charts can be used to analyze the performance of algorithms and determine Wait Time, Turnaround Time, and Response Time.

Given three processes with the following burst times,

P1	1
P2	2
P3	3

We have the following chart



From this we get wait times of

P1	0
P2	1
P3	3

The wait time and response time are identical.

We get turnaround times of:

P1 1
P2 3
P3 6

The average wait time is $\frac{4}{3}$ and the average turnaround time is $\frac{10}{3}$.