吳承瑀 110550091

OS Quiz

T/F (20%)

CLONE_FS (Files System)

F 1. If clone() is invoked with CLONE_FILE set, the parent and child tasks will share the same file-system information such as the current working directory.

F 2. Rate-monotonic scheduling schedules periodic tasks using a dynamic priority policy with preemption.

T 3. Asynchronous cancellation stops a thread immediately, even if it is in the middle of performing an update.

T 4. In a non-preemptive kernel system, a kernel-mode user process will run until it exits kernel mode, or voluntarily yields control of the CPU.

T 5. In Solaris's scheduling policy for time-sharing class, if a task has used up its entire time quantum without blocking, it will get a larger time quantum next time. Because its priority is lowered -> larger time quantum

F 6. Using one-to-one thread model, a process cannot take advantage of multiple processors by creating more user threads

T 7. A non-preemptive kernel is essentially free from race conditions on kernel data structures.

F 8. Tread-local storage is a shared memory allowing threads (belonging to the same process) to communicate.

F 9. In multi-processor systems, push (or pull) migration can be used to improve processor affinity.

F 10. Aging is a technique to prevent Convoy effect in CPU scheduling.

Multiple Choices

BCE 1. (6%) Please mark the algorithms listed below that could result in starvation. You can assume that each process will use the CPU for a finite burst before performing I/O.

(a) FCFS [✓ (b) SJF] [✓ (c) SRTF] (d) RR [✓ (e) Priority (preemptive)] ~~(f)~~ proportional share scheduling

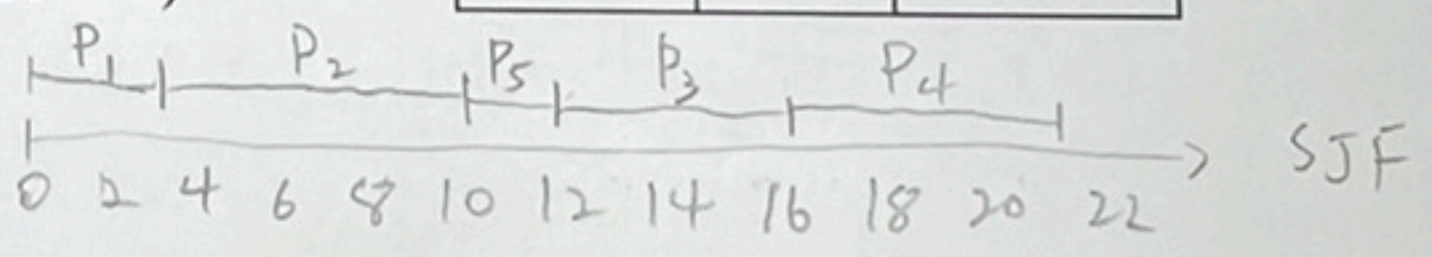
Short Answer

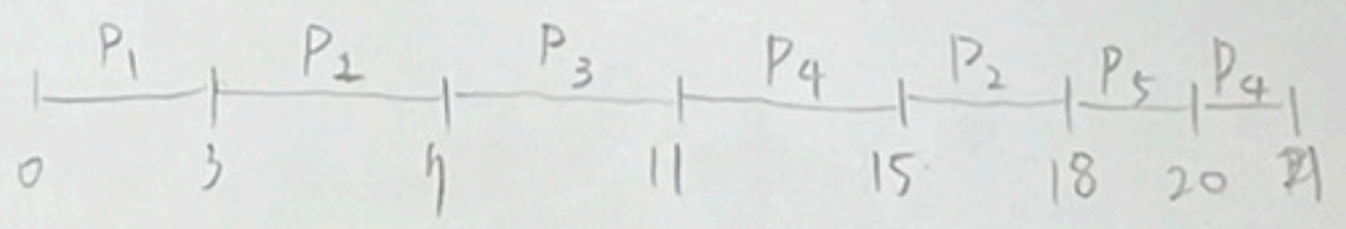
1. (27%) Here is a table of processes and their associated arrival and running times. For each scheduling algorithm below, indicate the *average waiting time* (WT), *average response time* (RT), and *average turn-around time* (TRT), respectively. Assume that the context switch overhead is 0

Process	Arrival	CPU burst
P1	0	3
P2	2	7
P3	4	4
P4	6	5
P5	8	2

(a) SJF (non-preemptive) (b) SRTF / (c) RR (time slice = 4)

Algorithms	Avg. WT	Avg. RT	Avg. TRT
SJF	4.2	4.2	8.4
SRTF	3.2	1	7.4
RR	6.4	3.8	10.6





2. (23%) Given three periodic processes with their periods and processing time (in terms of ms), please answer questions below.
(a) (6%) Which process(es) will run during 60~70ms if RMS is used?
(b) (12%) Which process(s) is scheduled to run during 75~80ms, and 115~120ms when EDF scheduling algorithm is used?
(c) (5%) Will any process miss its deadline within the first 150ms if (i) RMS (ii) EDF is used, respectively ?

Process	T (CPU burst)	P (Period)
P1	20	35
P2	15	40
P3	5	50

10
0 9 3 ~~12~~ 10
0 1 3 5 ~~4~~ 10
3 16

(a) 60~70: P_2	(b) 75~80: P_3 115~120: P_1		(c) RMS: Yes, P_3 will be missed EDF: No

3. (12%) Consider the two code segments, (a) and (b), below. What's the output for each code segment? (Assume all the processes and threads are created successfully. Please display your answer in the correct output order if there is more than one output, and please list all the possible results if there is more than one possibility of output order or output values.

Code (a)	Code (b)
#include <pthread.h>	#include <pthread.h>
#include <stdio.h>	#include <stdio.h>
int value = 5;	int value = 5;
main() {	void *runner(void *param) {
pid_t pid;	value += 20;
pid = **fork()**;	pthread_exit(0);
if(pid == 0) {	}
value += 20;	main() {
}	pid_t pid;
elseif(pid > 0) {	pthread_t tid;
~~wait(NULL);~~	pthread_attr_t attr;
}	pthread_attr_init(&attr);
printf("value: %d", value);	**pthread_create**(&tid, &attr, runner, NULL);
}	~~pthread_join(tid, NULL);~~
	printf("value: %d", value);
output: value =25	} output: value = 25
value =5	

4. (12%) Please answer the same question as above (question ?) except that the **wait()** in code (a) is removed and the **pthread_join()** in code (b) is removed.

(a) ① output: value = 25
value = 5

② output: value = 5
Value = 25

(b) ① output: value = 25
value = 5

② output: value = 5
value = 25

5
or
25