LUT University

Operating Systems and System Programming

LAND
OF THE
CURIOUS

Experiment 12

- Assignment 1-8
- This week we will not have new exercises and continue to work on the projects.

Experiment 12

- A2Q2.Describe the main differences between processes and programs.
- 1) Programs are permanent; processes are temporary. That is, the process has a life cycle, creation, execution, cancellation, etc.
- 2) The program is static and the process is dynamic;
- 3) A program can correspond to multiple processes; a process can execute one program or multiple programs
- 4) The process has concurrency, but the program does not;
- 5) A process is a basic unit that can run independently, allocate resources independently and accept scheduling independently, but a program is not.

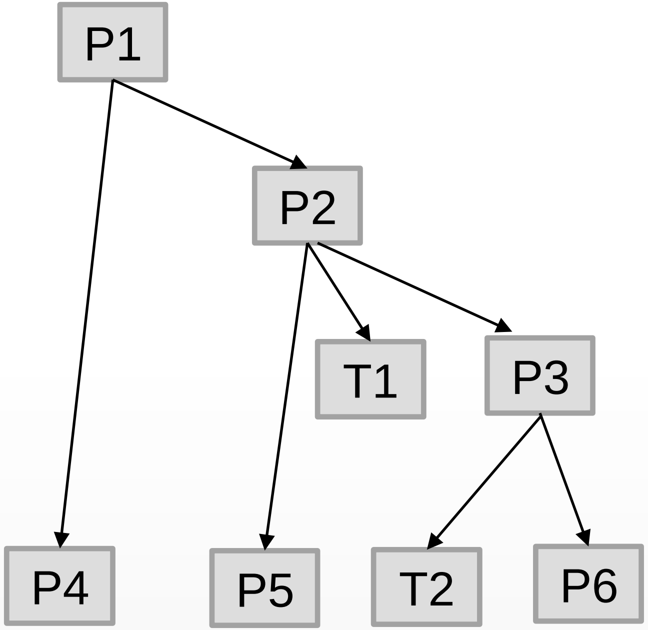
Experiment 12

- A3Q6
- The following program uses the Pthreads API. What would be the output from the program at LINE C and LINE P?
- int value = 0;
- void *runner(void *param); /* the thread */
- int main(int argc, char *argv[]){
- pid_t pid;
- pthread_t tid;
- pthread_attr_t attr;
- pid = fork();
- if (pid == 0) { /* child process */
- pthread_attr_init(&attr);
- pthread_create(&tid,&attr,runner,NULL);

pthread_join(tid,NULL);
printf("CHILD: value = %d",value); /* LINE C */
}
else if (pid > 0) { /* parent process */
wait(NULL);
printf("PARENT: value = %d",value); /* LINE P */
}}
void *runner(void *param) {
value = 5;
pthread_exit(0);}

Experiment 12

- A3Q7. Consider the following code segment.
- pid t pid;
- pid = fork();
- if (pid == 0) { /* child process */
-     fork();
-     thread create(. . .);
- }
- fork();
- a. How many unique processes are created? 6
- b. How many unique threads are created? 2

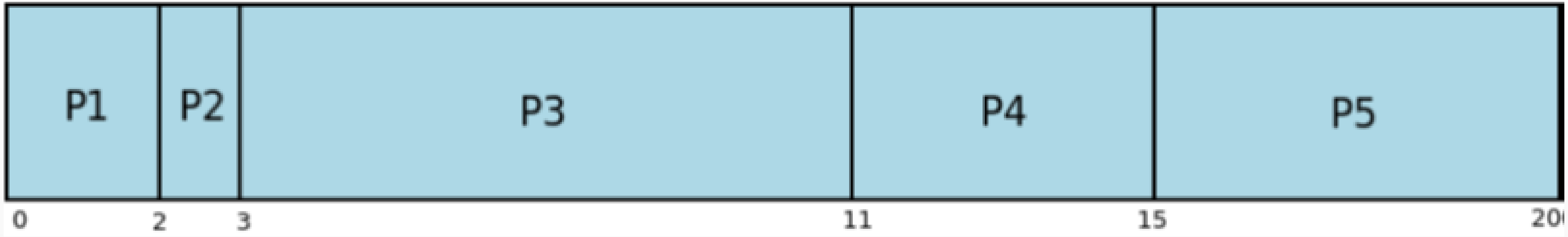


Experiment 12

- A5Q3
- Consider the following set of processes, with the length of the CPU burst time given in milliseconds:
- The processes are assumed to have arrived in the order P1, P2, P3, P4, P5, all at time 0.
- a. Draw four Gantt charts that illustrate the execution of these processes using the following scheduling algorithms: FCFS, SJF, non-preemptive priority (a larger priority number implies a higher priority), and RR (quantum = 2).
- b. What is the turnaround time of each process for each of the scheduling algorithms in part a?
- c. What is the waiting time of each process for each of these scheduling algorithms?
- d. Which of the algorithms results in the minimum average waiting time (over all processes)?

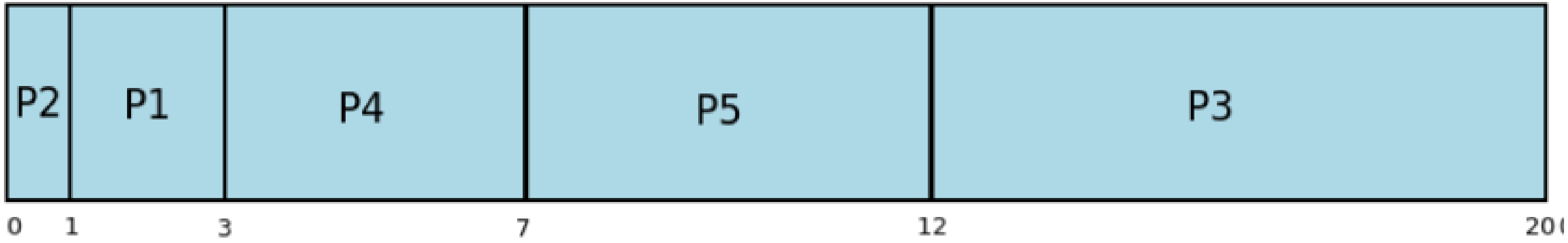
Experiment 12

Process	Burst Time	Priority
P_1	2	2
P_2	1	1
P_3	8	4
P_4	4	2
P_5	5	3

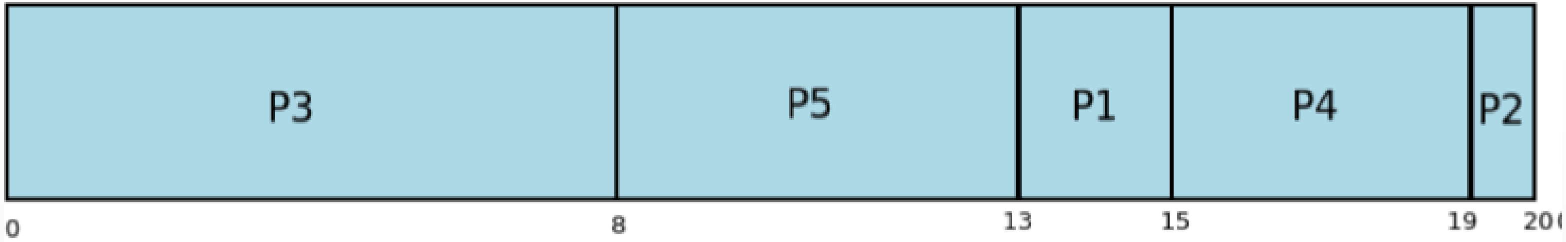


FCFS

Experiment 12

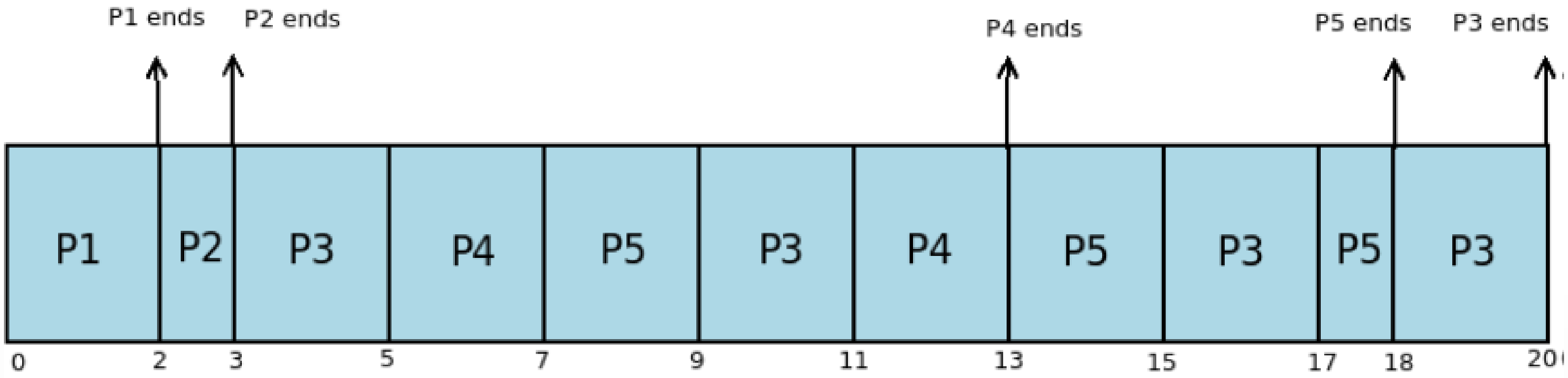


SJF



Priority

Experiment 12



RR

Experiment 12

- A5Q4
- The following processes are being scheduled using a preemptive, round-robin scheduling algorithm.
- Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. In addition to the processes listed below, the system also has an idle task (which consumes no CPU resources and is identified as Pidle). This task has priority 0 and is scheduled whenever the system has no other available processes to run. The length of a time quantum is 10 units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.
- a. Show the scheduling order of the processes using a Gantt chart.
- b. What is the turnaround time for each process?
- c. What is the waiting time for each process?
- d. What is the CPU utilization rate?

Experiment 12

Process	Priority	Burst	Arrival
P_1	40	20	0
P_2	30	25	25
P_3	30	25	30
P_4	35	15	60
P_5	5	10	100
P_6	10	10	105

a.

P_1	P_1	P_{idle}	P_2	P_3	P_2	P_3	P_4	P_4	P_2	P_3	P_{idle}	P_5	P_6	P_5
10	20	25	35	45	55	60	70	75	80	90	100	105	115	120

Experiment 12

- A6Q4
- Consider the following snapshot of a system:

	Allocation	Max	Available
	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	2 2 2 4
T_1	2 1 0 2	4 2 3 2	
T_2	2 4 1 3	2 5 3 3	
T_3	4 1 1 0	6 3 3 2	
T_4	2 2 2 1	5 6 7 5	

Experiment 12

- Answer the following questions using the banker's algorithm:
- a. Illustrate that the system is in a safe state by demonstrating an order in which the threads may complete.
- b. If a request from thread T4 arrives for (2, 2, 2, 4), can the request be granted immediately?
- c. If a request from thread T2 arrives for (0, 1, 1, 0), can the request be granted immediately?
- d. If a request from thread T3 arrives for (2, 2, 1, 2), can the request be granted immediately?

Experiment 12

	Allocation	Max	Available	Need
	A B C D	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	2 2 2 4	3 3 3 2
T_1	2 1 0 2	4 2 3 2		2 1 3 0
T_2	2 4 1 3	2 5 3 3		0 1 2 0
T_3	4 1 1 0	6 3 3 2		2 2 2 2
T_4	2 2 2 1	5 6 7 5		3 4 5 4

One possible order: T2,T0,T1,T3,T4

Experiment 12

- A request from thread T4 arrives for (2, 2, 2, 4)

	Allocation	Max	Available	Need
	A B C D	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	0 0 0 0	3 3 3 2
T_1	2 1 0 2	4 2 3 2		2 1 3 0
T_2	2 4 1 3	2 5 3 3		0 1 2 0
T_3	4 1 1 0	6 3 3 2		2 2 2 2
T_4	4 4 4 5	5 6 7 5		1 2 3 0

Experiment 12

- A request from thread T2 arrives for (0, 1, 1, 0)

	Allocation	Max	Available	Need
	A B C D	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	2 1 1 4	3 3 3 2
T_1	2 1 0 2	4 2 3 2		2 1 3 0
T_2	2 5 2 3	2 5 3 3		0 0 1 0
T_3	4 1 1 0	6 3 3 2		2 2 2 2
T_4	2 2 2 1	5 6 7 5		3 4 5 4

Experiment 12

- A request from thread T3 arrives for (2, 2, 1, 2)

	Allocation	Max	Available	Need
	A B C D	A B C D	A B C D	A B C D
T_0	3 1 4 1	6 4 7 3	0 0 1 2	3 3 3 2
T_1	2 1 0 2	4 2 3 2		2 1 3 0
T_2	2 4 1 3	2 5 3 3		0 1 2 0
T_3	6 3 2 2	6 3 3 2		0 0 1 0
T_4	2 2 2 1	5 6 7 5		3 4 5 4

Experiment 12

- A7Q1. Given five memory partitions of 100 KB, 500 KB, 200 KB, 300 KB, and 600 KB (in order), how would each of the first-fit, best-fit, and worst-fit algorithms place processes of 212 KB, 417 KB, 112 KB, and 426 KB (in order)? Which algorithm makes the most efficient use of memory?
- First-fit:
  - 212K is put in 500K partition (new partition 288K = 500K − 212K)
  - 417K is put in 600K partition
  - 112K is put in 288K partition
  - 426K must wait

Experiment 12

- Best-fit:
  - 212K is put in 300K partition
  - 417K is put in 500K partition
  - 112K is put in 200K partition
  - 426K is put in 600K partition
- Worst-fit:
  - 212K is put in 600K partition
  - 417K is put in 500K partition
  - 112K is put in 388K partition
  - 426K must wait

Experiment 12

- A8Q2. The following page table is for a system with 16-bit virtual and physical addresses and with 4,096-byte pages. The reference bit is set to 1 when the page has been referenced. A dash for a page frame indicates the page is not in memory. All numbers are provided in decimal.
- Convert the following virtual addresses (in hexadecimal) to the equivalent physical addresses. You may provide answers in either hexadecimal or decimal.

Experiment 12

Page	Page Frame	Reference Bit
0	9	0
1	1	0
2	14	0
3	10	0
4	–	0
5	13	0
6	8	0
7	15	0
8	–	0
9	0	0
10	5	0
11	4	0
12	–	0
13	–	0
14	3	0
15	2	0

$4096=2^{12}$

$16-12=4$

0xE12C:

E(14) $\rightarrow$ 3

312C

Experiment 12

- A8Q3. Assume we have a demand-paged memory. The page table is held in registers. It takes 8 milliseconds to service a page fault if an empty page is available or the replaced page is not modified, and 20 milliseconds if the replaced page is modified. Memory access time is 100 nanoseconds.
- Assume that the page to be replaced is modified 70 percent of the time. What is the maximum acceptable page-fault rate for an effective access time of no more than 200 nanoseconds?
- 0.2 μsec >= (1 − P) × 0.1 μsec + (0.3P) × 8 millisec + (0.7P) × 20 millisec

Experiment 12

- A8Q4. A page-replacement algorithm should minimize the number of page faults. We can achieve this minimization by distributing heavily used pages evenly over all of memory, rather than having them compete for a small number of page frames. We can associate with each page frame a counter of the number of pages associated with that frame. Then, to replace a page, we can search for the page frame with the smallest counter.
- a. Define a page-replacement algorithm using this basic idea. Specifically address these problems:
- i. What the initial value of the counters is
- ii. When counters are increased
- iii. When counters are decreased
- iv. How the page to be replaced is selected

Experiment 12

- Define a page-replacement algorithm addressing the problems of:
- i. Initial value of the counters—0.
- ii. Counters are increased—whenever a new page is associated with that frame.
- iii. Counters are decreased—whenever one of the pages associated with that frame has been replaced and is no longer required.
- iv. How the page to be replaced is selected—find a frame with the smallest counter. Use FIFO for breaking ties.

Experiment 12

- How many page faults occur for your algorithm for the following reference string, for four page frames?
- 1, 2, 3, 4, 5, 3, 4, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2.

1	2	3	4	5	3	4	1	6	7	8	7	8	9	7	8	9	5	4	5	4	2
1^1	1^1	1^1	1^1	5^2	5^2	5^2	5^2	5^2	5^2	5^2	5^2	5^2	5^2	5^2	8^3	8^3	8^2	8^2	8^2	8^2	8^2
	2^1	2^1	2^1	2^1	2^1	2^1	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	1^2	5^2	5^2	5^2	5^2	5^2
		3^1	3^1	3^1	3^1	3^1	3^1	6^1	6^1	8^1	8^1	8^1	9^2	9^2	9^2	9^2	9^2	9^2	9^2	9^2	9^2
			4^1	4^1	4^1	4^1	4^1	4^1	7^2	7^2	7^2	7^2	7^2	7^2	7^2	7^2	7^2	4^1	4^1	4^1	2^1