



東南大學  
SOUTHEAST UNIVERSITY

# OPERATING SYSTEM CONCEPTS

.....

Exercises

A/Prof. Kai Dong



# Contents

1. Process API
2. Thread API
3. Simple Scheduling Algorithms
4. Multi-Level Feedback Queue
5. Banker's Algorithm
6. Paging
7. Page Replacement
8. Disk Scheduling

Q



```
1  int value = 5;
2  int main(int argc, char *argv[]) {
3      pid_t pid;
4      pid = fork();
5      if (pid == 0) {
6          printf("child process, value1 : %d\n", value);
7          value += 15;
8          printf("child process, value2 : %d\n", value);
9      }
10     else if (pid > 0) {
11         printf("parent process, value3 : %d\n", value);
12         wait(NULL);
13         printf("parent process, value4 : %d\n", value);
14     }
15     exit(0);
16 }
```

- What is the output?

A



```
1 child process, value1 : 5
2 child process, value2 : 20
3 parent process, value3 : 5
4 parent process, value4 : 5
5
6 parent process, value3 : 5
7 child process, value1 : 5
8 child process, value2 : 20
9 parent process, value4 : 5
```

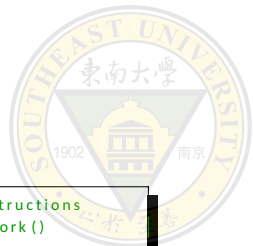
Q



- Including the initial parent process, how many processes are created by the following program.

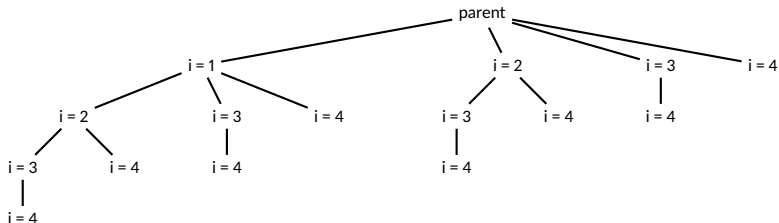
```
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main() {
5      int i;
6      for (i = 0; i < 4; i ++){
7          fork();
8      }
9  }
```

A



```
1  int i;  
2  for (i = 0; i < 4; i ++)  
3      fork();  
4  return 0;
```

```
1  0x1024 ...; some instructions  
    implementing fork()  
2  0x2028 incl %eax  
3  0x202c cmpl $0x0004, %eax  
4  0x2030 jne 0x1024  
5  0x2032 retn
```



Q



- Draw the diagram of process state.



# Contents

1. Process API
2. Thread API
3. Simple Scheduling Algorithms
4. Multi-Level Feedback Queue
5. Banker's Algorithm
6. Paging
7. Page Replacement
8. Disk Scheduling





## In Class Exercise

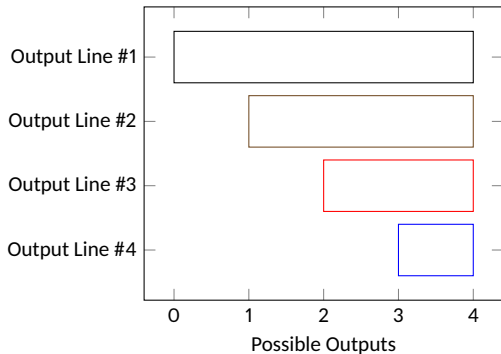
```
1  /* kai.c */
2  #include <stdio.h>
3  #include <pthread.h>
4  void *helloFunc(void *ptr) {
5      int *data;
6      data = (int *) ptr;
7      printf("I'm Thread %d\n", *data);
8      return (void *) data;
9  }
10 int main(int argc, char *argv[]) {
11     pthread_t hThread[4];
12     int *rvals[4];
13     for (int i = 0; i < 4; i++)
14         pthread_create(&hThread[i], NULL, helloFunc, (void *) &i);
15     for (int i = 0; i < 4; i++) {
16         pthread_join(hThread[i], (void **) &rvals[i]);
17         //printf("Thread %d returns %d\n", i, *rvals[i])
18     }
19     return 0;
20 }
```

```
1  prompt> gcc -o kai kai.c -pthread -Wall
2  prompt> ./kai
```

A



Any possible order of a non-decreasing sequence satisfying ...





# Contents

1. Process API
2. Thread API
- 3. Simple Scheduling Algorithms**
4. Multi-Level Feedback Queue
5. Banker's Algorithm
6. Paging
7. Page Replacement
8. Disk Scheduling

## Q

Consider the following set of processes, with the length of the CPU burst given in milliseconds:

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

The processes are assumed to have arrived in the order of  $P_1, P_2, P_3, P_4, P_5$ , all at time 0.

1. 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).
2. What is the average turnaround time for each of the scheduling algorithms in part 1?
3. What is the waiting time of each process for each of these scheduling algorithms?
4. Which of the algorithms results in the minimum average waiting time (over all processes)?



A



Q1:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
FCFS	P <sub>1</sub>		P <sub>2</sub>	P <sub>3</sub>								P <sub>4</sub>				P <sub>5</sub>				
SJF	P <sub>2</sub>	P <sub>1</sub>		P <sub>4</sub>				P <sub>5</sub>				P <sub>3</sub>								
PRIO	P <sub>3</sub>								P <sub>5</sub>				P <sub>1</sub>		P <sub>4</sub>			P <sub>2</sub>		
RR	P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>		P <sub>4</sub>		P <sub>5</sub>		P <sub>3</sub>		P <sub>4</sub>	P <sub>5</sub>		P <sub>3</sub>	P <sub>5</sub>	P <sub>3</sub>	P <sub>5</sub>	P <sub>3</sub>		

Q2:

	FCFS	SJF	PRIO	RR
$T_{\text{turnaround}} \text{ (ms)}$	$\frac{2+3+11+15+20}{5}$ =10.2	$\frac{1+3+7+12+20}{5}$ =8.6	$\frac{8+13+15+19+20}{5}$ =15	$\frac{2+3+20+13+18}{5}$ =11.2

Q3 &amp; Q4:

		P <sub>1</sub>	P <sub>2</sub>	P <sub>3</sub>	P <sub>4</sub>	P <sub>5</sub>	Avg
$T_{\text{waiting}} \text{ (ms)}$	FCFS	0	2	3	11	15	6.2
	SJF	1	0	12	3	7	4.8
	PRIO	13	19	0	15	8	11
	RR	0	2	12	9	13	7.2

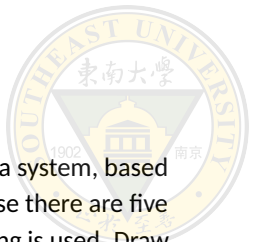
Minimum



# Contents

1. Process API
2. Thread API
3. Simple Scheduling Algorithms
- 4. Multi-Level Feedback Queue**
5. Banker's Algorithm
6. Paging
7. Page Replacement
8. Disk Scheduling

Q



Given the table below showing the process information of a system, based on multilevel feedback queuing scheduling scheme. Suppose there are five levels in the system and within each level the FCFS scheduling is used. Draw a Gantt chart to show the time of CPU allocated to each process until all processes are finished using time quantum  $q = 2^i$ , (where  $q$  = time allocated for a process to run in its turn, and  $i$  ranges from 0 to 4 indicating the  $i^{th}$  level queue).

Process	Arrival Time	Service Time
$P_1$	0	3
$P_2$	1	5
$P_3$	3	2
$P_4$	9	5
$P_5$	12	5

A



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$Q_0$	$P_1$	$P_2$		$P_3$						$P_4$			$P_5$							
$Q_1$			$P_1$		$P_1$	$P_2$	$P_3$				$P_4$			$P_5$						
$Q_2$									$P_2$							$P_2$	$P_4$		$P_5$	





# Contents

1. Process API
2. Thread API
3. Simple Scheduling Algorithms
4. Multi-Level Feedback Queue
- 5. Banker's Algorithm**
6. Paging
7. Page Replacement
8. Disk Scheduling

## Q

A computer system has 3 types of resources A, B, and C with different numbers of instances. There are 4 running processes  $P_1, P_2, P_3, P_4$ . The total resources, the resource's *Allocation* and *Max* matrices for the four processes are shown as follows:

Process	Allocation			Max		
	A	B	C	A	B	C
$P_1$	1	3	1	6	5	3
$P_2$	0	2	2	3	5	3
$P_3$	2	0	0	3	5	2
$P_4$	0	1	3	2	4	3
total	6	9	6			

1. What are the matrices *Need* and *Available* for the system?
2. Please check if the system is currently deadlocked. Show your steps clearly.
3. At the current state, if  $P_2$  requests additional resources  $[1, 0, 0]$ , can the request be granted without any possible deadlock? Show your steps clearly.
4. At the current state, if  $P_1$  requests additional resources  $[1, 0, 0]$ , can the request be granted without any possible deadlock? Show your steps clearly.

# A

- $n = 4, m = 3$ , Allocation and Max are already defined.

- Key to Q1: Compute Available and Need

Process	Need		
	A	B	C
$P_1$	5	2	2
$P_2$	3	3	1
$P_3$	1	5	2
$P_4$	2	3	0
Available	3	3	0

- Key to Q3 Q4: Resource-Request Alg.
- Assume the request is granted, use the Safety Alg. to determine whether the system is in safe state.

- Key to Q2: Safety Alg.

1.  $Work = Available$ ,  
 $Finish[i] = false, 0 \leq i < n$ .

2. Find an  $i = 4$ .

3.  $Work = Work + Allocation_i$ ,  
 $Finish[4] = true$

	A	B	C
Work	3	4	3

4. Go to step 2.

5. Find an  $i = 2$ .

6.  $Work = \langle 3, 6, 5 \rangle$ ,  
 $Finish[2] = true$

7. Go to step 2.

8. Find an  $i = 3$ .

- Safe sequence:  $\langle P_4, P_2, P_3, P_1 \rangle$



Q



Consider the following system snapshot using the data structures in the Banker's algorithm, with resources A, B, C, and D, and processes P0 to P4:

	Max				Allocation				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P0	2	3	3	3	1	2	1	2								
P1	1	4	1	0	1	1	0	0								
P2	2	1	1	1	0	1	0	1								
P3	5	4	3	3	1	1	2	2								
P4	4	2	6	3	1	2	1	2								
Total Res													2	0	2	0

1. How many resources of type A, B, C, and D are there?
2. What are the contents of the Need matrix?
3. Is the system in a safe state? Why?
4. If a request from process P2 arrives for additional resources of (0,0,2,0), can the Banker's algorithm grant the request immediately? Why?



# Contents

1. Process API
2. Thread API
3. Simple Scheduling Algorithms
4. Multi-Level Feedback Queue
5. Banker's Algorithm
- 6. Paging**
7. Page Replacement
8. Disk Scheduling



Consider a system with 64 MB of physical memory, 32-bit physical addresses, 32-bit virtual addresses, and 4 KB physical page frames.

- (a) Using a single-level paging scheme, what is the maximum number of page table entries for a page table in this system?
- (b) Using a two-level paging scheme with a 1024-entry outer-page table, how many bits are needed in the logical address to represent the outer page table? How many bits are needed in the logical address in order to represent the inner page table? How many bits are used to represent the offset within a page?
- (c) Suppose a TLB is used with the two-level paging scheme described in part (b), and the TLB has a 90% hit rate. If the TLB access time is 10 ns and memory access time is 100 ns, what is the effective memory access time of the system?

## A



Consider a system with 64 MB of physical memory, 32-bit physical addresses, 32-bit virtual addresses, and 4 KB physical page frames.

(a) Using a single-level paging scheme, what is the maximum number of page table entries for a page table in this system?

$2^{20}$

(b) Using a two-level paging scheme with a 1024-entry outer-page table, how many bits are needed in the logical address to represent the outer page table? How many bits are needed in the logical address in order to represent the inner page table? How many bits are used to represent the offset within a page?

10, 10, 12

(c) Suppose a TLB is used with the two-level paging scheme described in part (b), and the TLB has a 90% hit rate. If the TLB access time is 10 ns and memory access time is 100 ns, what is the effective memory access time of the system?

$EAT = 90\% \times (10 + 100) + 10\% \times (20 + 300) = 99 + 32 = 131$



# Contents

1. Process API
2. Thread API
3. Simple Scheduling Algorithms
4. Multi-Level Feedback Queue
5. Banker's Algorithm
6. Paging
- 7. Page Replacement**
8. Disk Scheduling



Q



Consider the reference page sequence is 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5, and the number of page frame is 3.

- (a) How many page faults for FIFO algorithm?
- (b) How many page faults for LRU algorithm?
- (c) How many page faults for OPT algorithm?

A

Consider the reference page sequence is 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5, and the number of page frame is 3.

(a) How many page faults for FIFO algorithm? Key: 9

	1	2	3	4	1	2	5	1	2	3	4	5
	1	1 2	1 2	4 2	4 1	4 1	5 1	5 1	5 1	5 3	5 3	5 3
PF	o	o	o	o	o	o	o			o	o	

(b) How many page faults for LRU algorithm? Key: 10

	1	2	3	4	1	2	5	1	2	3	4	5
	1	1 2	1 2	4 2	4 1	4 1	5 1	5 1	5 1	3 1	3 4	3 4
PF	o	o	o	o	o	o	o			o	o	o

(c) How many page faults for OPT algorithm? Key: 7

	1	2	3	4	1	2	5	1	2	3	4	5
	1	1 2	1 2	1 2	1 2	1 2	1 2	1 2	1 2	3 2	4 2	4 2
PF	o	o	o	o			o			o	o	





# Contents

1. Process API
2. Thread API
3. Simple Scheduling Algorithms
4. Multi-Level Feedback Queue
5. Banker's Algorithm
6. Paging
7. Page Replacement
8. Disk Scheduling

## Q



Suppose that a disk drive has 200 cylinders, numbered 0 to 199. The drive is currently serving a request at cylinder 123, and the previous request was at cylinder 175. The queue of pending requests, in FIFO order, is 86, 147, 180, 28, 95, 151, 12, 77, 30. Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms respectively?

1. FCFS
2. SSTF
3. C-SCAN
4. LOOK