University of Stuttgart
Institute of Parallel and Distributed Systems
Distributed Systems Department

Winter Term 2017/2018

Operating Systems

Assignment Sheet 3

Available:	Monday, November 20, 2017, 08:00 am
Due Date:	Monday, December 04, 2017, 08:00 am
Discussion:	Friday, December 08, 2017, 11:30-13:00, Room V38.02

General remarks: The intent of the exercises is to help you get more familiar with fundamental concepts of modern operating systems as presented in the lecture.

Work on the problems with your group and hand in a single solution which will be graded for all members of the group. We will review your solution and hand out the graded version in the exercise sessions. Everybody has to present at least once in the sessions. If you hand in a solution but are not able to present it, no points will be given *for the whole exercise*! If we detect any plagiarism in your solution, *the whole group* will not get points *for the whole sheet.*

Please read the *SubmissionGuidlines.pdf* and *InformationOnExercises.pdf* in ILIAS in the Course Material folder. Especially, **be concise** and avoid long answers.

For additional reference, consult the text books cited in the lecture and explore the Web. You can find additional information on lectures and exercises at the lecture homepage. If you have any questions, you may contact us on the e-learning system ILIAS.

Questions

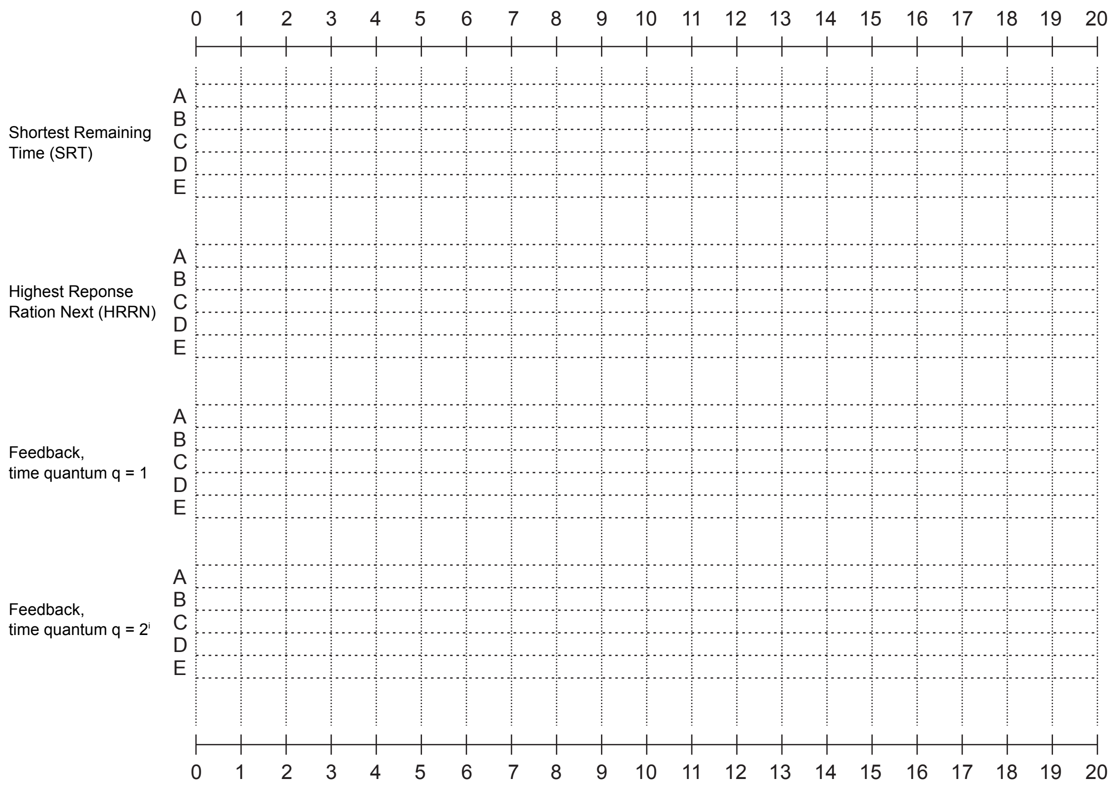
1. (6 points) Uniprocessor Scheduling II
   (a) Consider the following set of jobs:

Job Name	Arrival Time	Service Time (T_s)
A	0	4
B	2	7
C	4	2
D	5	4
E	7	1

   For each of the following scheduling policies, draw a Gantt diagram (timing diagram) that illustrates which job is allocated the processor at each time unit. You may use the grid on the next page. (4 points)
   i. Shortest remaining time scheduling
   ii. Highest response ratio next scheduling
   iii. Feedback scheduling, time quantum $q = 1$ (Assume five queues Q0-Q4)
   iv. Feedback scheduling, time quantum $q = 2^i$ (Assume five queues Q0-Q4)

   (b) The ratio of turnaround time (T_r) and service time (T_s) is a measure of the *relative delay* that a job has to experience during its complete execution. By filling the table on the next page for each scheduling scheme, state which scheduling scheme has the best *average* response time and which one provides least *average* relative delay, T_r/T_s, for the given job set.

   For determining response time, you may assume that one quantum of execution time is enough for every job to generate a response. (2 points)



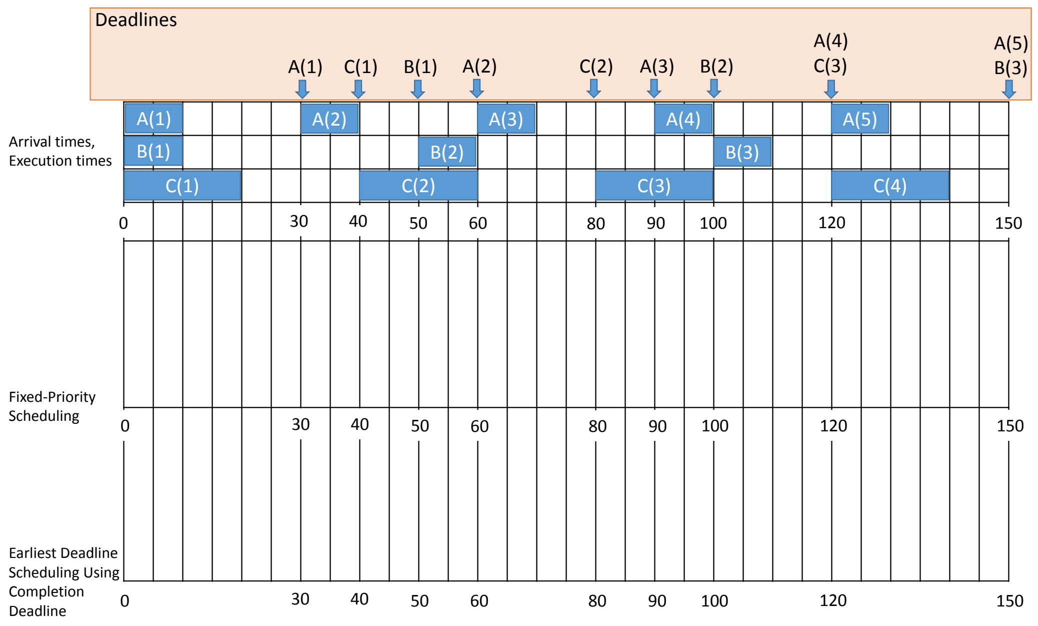
	Process	A	B	C	D	E	Mean
	Arrival Time	0	2	4	5	7	
	Service Time (Ts)	4	7	2	4	1	
Shortest Remaining Time (SRT)	Response Time						
	Turnaround Time (Tr)						
	Relative Delay (Tr/Ts)						
Highest Response Ratio Next (HRRN)	Response Time						
	Turnaround Time (Tr)						
	Relative Delay (Tr/Ts)						
Feedback time Quantum q=1	Response Time						
	Turnaround Time (Tr)						
	Relative Delay (Tr/Ts)						
Feedback Time Quantium q=2^i	Response Time						
	Turnaround Time (Tr)						
	Relative Delay (Tr/Ts)						

2. (7 points) Real-Time Scheduling

(a) Consider a set of three periodic tasks with the following execution profiles:

Process	**Arrival Time**	**Execution Time**	**Ending Deadline**
A(1)	0	10	30
A(2)	30	10	60
• • •			
B(1)	0	10	50
B(2)	50	10	100
• • •			
C(1)	0	20	40
C(2)	40	20	80
• • •			

Complete the scheduling diagram shown below using Fixed-priority and Earliest-Deadline-First scheduling schemes. Assume that for Fixed-priority scheduling, process B has highest priority, and process C has higher priority than process A. Indicate which deadlines cannot be met, if any. (2 points)



(b) Which of the two scheduling schemes in part (a) causes greater overhead for the Dispatcher. Justify your answer. (1 point)

(c) Consider a task set with the following independent periodic tasks:

- Task P_1: $C_1 = 20$; $T_1 = 60$
- Task P_2: $C_2 = 40$; $T_2 = 100$

Can these tasks be successfully scheduled using rate monotonic scheduling? Justify your answer with a schedulability test. (1 points)

(d) Now add the following task to the set of tasks given in part (c):

- Task P_3: $C_3 = 20$; $T_3 = 120$

Is schedulability test for rate monotic scheduling satisfied? (1 points)

(e) Suppose that the first instance of each of the preceding three tasks from part (c) and (d) arrives at time $t = 0$. Assume that the first deadline for each task is the following:

- $D_1 = 60$
- $D_2 = 100$
- $D_3 = 120$

Using rate monotonic scheduling, will all three deadlines be met? What about deadlines for future repetitions of each task? What conclusion do you draw about your result in part (d)? (2 points)

3. (7 points) Multiprocessor Scheduling

Consider a multiprocessor system with 4 CPUs that has to schedule the following processes with their respective threads:

Process	Thread	Arrival Time	Execution Time
A	A_0	0	3
A	A_1	0	3
B	B_0	1	5
B	B_1	1	5
C	C_0	2	3
C	C_1	2	3
C	C_2	2	5
C	C_2	2	5
D	D_0	3	4
D	D_1	3	2
E	E_0	7	4
E	E_1	7	4
E	E_2	7	4
E	E_3	7	4

(a) Schedule the specified processes, i.e. their threads, by employing the *Load Sharing* scheme presented in the lecture. The threads are scheduled according to FIFO order in the ready queue (non preemptive). For the threads belonging to the same process, the order of arrival is decided by the index in their subscript, for example C_1 would be scheduled after C_0 etc. Put your result in a tabular form similar to the one shown below. (2 points)

time	CPU 1	CPU 2	CPU 3	CPU 4
0				
1				
2				
.				
.				

(b) Another scheduling approach for multiprocessor systems explained in the lecture was *Space Sharing*. Using *Space Sharing*, schedule the specified set of processes, i.e. their threads, and present your results in a tabular form as done in part (a). Note that the size of a partition (number of CPUs) required for scheduling a particular process is determined by the number of its threads. A CPU is free only after it completes the full execution of the thread that was assigned to it. (2 points)

(c) Consider two processes A and B with their threads (A_0, A_1, B_0, B_1) which are to be scheduled on a system with 2 CPUs. The threads A_0 and A_1 represent a Request-Response relationship and therefore can effect the execution time of each other. Assume that a thread executes one line of code in one time unit. (3 points)

Process A			
integer $x \leftarrow 0, y \leftarrow 0$			
A_0		A_1	
1:	**while** $y = 0$ **do**	1:	$y \leftarrow 1$
2:	;	2:	**while** $x = 0$ **do**
3:	$x \leftarrow 1$	3:	;

Process B			
integer $i \leftarrow 1, j \leftarrow 1$			
B_0		B_1	
1:	**while** $i < 5$ **do**	1:	**while** $j < 10$ **do**
2:	$i \leftarrow i + 1$;	2:	$j \leftarrow j + 1$;

i. Briefly state an execution scenario in which the total processing time of both processes reaches its maximum value.

ii. Briefly state an execution scenario in which the total processing time of both processes reaches its minimum value.

iii. Which scheduling scheme from parts (a) and (b) **guaranties** the minimum total processing time? Justify your answer.

4. (7 points) Implementation Task

   In this task, you will further extend the shell, as developed in Exercise Sheet 1 & 2, by a few more advanced features.

   (a) (2 points) Implement background execution of programs. An ampersand (`&`) at the end of the command line indicates that the shell should return to the command line prompt immediately after launching that program.

   Hint: The solution involves the **fork** function.

   (b) (5 points) Programs can be run together such that one program reads the output from another with no need for an explicit intermediate file. In the following line,

   ```
   <cmd1> | <cmd2>
   ```

   command `<cmd1>` is executed, and its output is used as the input of `<cmd2>`. This is commonly called piping, since the `|` character is known as a “pipe”. Pipes are an approach for inter-process communication by message passing, besides shared memory as discussed in the lecture.

   i. Add support for pipes to `gbsh`.

   ii. Also allow the chaining of multiple pipes.

   Hint: To do this, you’ll need to use the **pipe** and the **dup2** system calls.