

# System Programming

Programming assignment 3

陳心萍  
王 俊智  
廖冠豪



# Task

- Deal with two scheduling problems between **bidding\_system** and **customers** using **signals**.
- One is **priority scheduling**, while the other called **earliest deadline first scheduling**.
- You have to implement following program:
  - bidding\_system.c
  - customer.c (optional)
  - bidding\_system\_EDF.c



# Priority scheduling

- There are three types of customer:
  - Ordinary customer
  - Member customer
  - VIP customer
- The priority order of customer is  $VIP > member > ordinary$ .
- A customer with **higher priority** should be dealt with **first** when **bidding\_system receives** him/her.



# Priority scheduling

- Example 1:

If an **ordinary customer c2** arrives at the time the bidding\_system deals with an **ordinary customer c1**, the bidding\_system should **finish c1 first**.

- Example 2:

If a **VIP customer c2** arrives at the time the bidding\_system deals with an **member customer c1**, the bidding\_system should **stop to deal with the VIP customer c2**.

- Example 3:

If an **member customer** or an **ordinary customer c2** arrives at the time the bidding\_system deals with a **VIP customer c1**, the bidding\_system should finish a **VIP customer c1 first**.



# Priority scheduling

- Customer information:

Table 1.1: customer information

customer code	type	process time(sec)	limit time(sec)
0	ordinary	1.0	no limit
1	member	0.5	1.0
2	VIP	0.2	0.3

[\*] You can use for loop and nanosleep() to simulate the time consuming, but you should notice that nanosleep() may be interrupted by signal.

[\*] If **customer** sends a VIP customer at 8:01:53.1, the customer should be sent back before 8:01:53.4



# Structure of program - bidding\_system (1)

- bidding\_system:  
*./bidding\_system [test\_data]*
- Bidding\_system should **fork** one child and **execute customer** program. Bidding\_system should use **pipe** to receive **ordinary** customer and redirect customer's stdin/stdout to pipe.
- There is a file “**bidding\_system\_log**” record the status of bidding\_system.



# Structure of program - bidding\_system (2)

- When bidding\_system receives and **starts dealing with the customer**, it needs to write  
*receive [customer code] [serial number]*  
to log file.
- When bidding\_system finished him/her, it needs to write  
*finish [customer code] [serial number]*  
to log file and send a signal to customer.
- When bidding\_system read “**EOF**” from pipe, it should write “**terminate  
\n**” to file “bidding\_system\_log” and terminate the program itself.
- If no customer blocked, the bidding\_system needs to wait until next customer arriving)



## Structure of program - bidding\_system (3)

- Bidding\_system send signal to tell sender that it finish process a customer.

Type	Signal
Ordinary	SIGINT
Member	SIGUSR1
VIP	SIGUSR2



# Structure of program - customer (1)

- Customer (optional)

*./customer [test\_data]*

- Customer reads [test file] and **send signal** to bidding\_system at **specific time**.
- There is a file record the status of customer called “**customer\_log**”.



## Structure of program - customer (2)

- When customer **sends** the customer, it needs to write

*send [customer code] [serial number]*

to file “customer\_log”.

- When customer gets the finished customer(**a specific signal**) sent back by bidding\_system, it needs to write

*finish [customer code] [serial number]*

to the file.

- If bidding\_system don't send back the customer **within the specific time**, the customer should write

*timeout [customer code] [serial number]*

to the file “customer\_log” and **terminate the program** itself.



## Structure of program - customer (3)

- Customer send signal to tell bidding\_system that it have upcoming customer to process.

Type	How to send to bidding_system
Ordinary	ordinary\n
Member	SIGUSR1
VIP	SIGUSR2

[\*] In our test data, we assure that if **customer** sent a member customer, then it would not send another member customer before receiving the previous member customer(for avoiding signal missing). And so is the case of VIP customers.



# Format of input and output (1)

- test\_data

*[customer code] [sending time]*

- Each line has two numbers:
  - The first number is a **customer code** corresponding to **each type of customer**.
  - The second number is the **sending time** of the customer. The time is in **ascending order** in the test\_data.



# Format of input and output (2)

- bidding\_system\_log
- [action] [customer code] [serial number]
  - [action] will be “receive” or “finish”.
  - [customer code] will be 0, 1, or 2.
  - [serial\_number] is the serial number of customer.



# Format of input and output (2)

- customer\_log
- [action] [customer code] [serial number]
  - [action] will be “receive”, “finish” or “timeout”.
  - [customer code] will be 0, 1, or 2.
  - [serial\_number] is the serial number of customer.

[\*] In our test data, we assure that all actions will not at the same time. That is, if a customer is finished at 8:01:53.1, no signal will be sent at 8:01:53.1.



# Sample Execution

test\_data

0 0

1 0.8

2 1.2

2 2.1

1 2.2



# Sample Execution

bidding\_system\_log

receive 0 1

receive 1 1

receive 2 1

finish 2 1

finish 1 1

finish 0 1

receive 2 2

finish 2 2

receive 1 2

finish 1 2

terminate



# Sample Execution

customer\_log

send 0 1

send 1 1

send 2 1

finish 2 1

finish 1 1

finish 0 1

send 2 2

send 1 2

finish 2 2

finish 1 2



# Earliest deadline first scheduling

- There are three types of customer, **ordinary** customer, **patient** customer and **impatient** customer.
- A customer's deadline with the following formula:  
$$deadline = arrive\ time + limit\ time$$
- The customer with **earliest deadline** should be **dealt with first**.
- Customer information is show as follow:

Table 1.2: customer information

customer code	type	process time(sec)	limit time(sec)
0	ordinary	0.5	2.0
1	patient	1.0	3.0
2	impatient	0.2	0.3



# Structure of program - EDF (1)

- bidding\_system\_EDF  
*./bidding\_system\_EDF [test\_data]*
- customer\_EDF  
*./customer\_EDF [test\_data]*
- The action of bidding\_system\_EDF and customer\_EDF is same as priority version.
- The difference between two scheduling is **communication** between bidding\_system and customer\_EDF.



## Structure of program - EDF (2)

- Both bidding\_system\_EDF and customer\_EDF send same signal to specific customer type.

Type	Signal
Ordinary	SIGUSR1
Patient	SIGUSR2
Impatient	SIGUSR3

[\*] Please add `#define SIGUSR3 SIGWINCH` to your code.



## Structure of program - EDF (3)

- When bidding\_system\_EDF receives “**terminate  
\n**” via pipe, it should write “**terminate\n**” to file “bidding\_system\_log” and terminate the program itself.



# Format of input and output (EDF)

- The format of test\_data, bidding\_system\_log, customer\_log are **the same as** priority scheduling version.



# Sample Execution - EDF

test\_data

0 0

1 0.4

2 0.8

2 2.1

1 2.2

0 2.6



# Sample Execution - EDF

bidding\_system\_log

receive 0 1

finish 0 1

receive 1 1

receive 2 1

finish 2 1

finish 1 1

receive 2 2

finish 2 2

receive 1 2

receive 0 2

finish 0 2

finish 1 2

terminate



# Sample Execution - EDF

customer\_log

send 0 1

send 1 1

finish 0 1

send 2 1

finish 2 1

finish 1 1

send 2 2

send 1 2

finish 2 2

send 0 2

finish 0 2

finish 1 2



# Grading

- There are 6 subtasks in this assignment, you can get 8 points if you finish all of them.
- Overall
  - ✦ (1pt) Produce executable files successfully. Your Makefile can generate bidding\_system and bidding\_system\_EDF and customer.
- Priority Scheduling
  - ✦ (1pt) Your bidding\_system can deal with ordinary customers.
  - ✦ (1pt) Your bidding\_system can deal with ordinary customers and member customers.
  - ✦ (1pt) Your bidding\_system can deal with all types of customers. (ordinary, member, VIP)
  - ✦ (2pt) You implement customer by yourself. You only need to implement customer based on priority scheduling (which would terminate if timeout)



# Grading (2)

- Earliest deadline first scheduling
  - ♦ (1pt) Your bidding\_system\_EDF can deal with ordinary customers and patient customers.
  - ♦ (1pt) Your bidding\_system\_EDF can deal with all types of customers. (ordinary, patient, impatient)



# Submission

- Submit `SP_HW3_{student_id}.tar.gz` to CEIBA before the deadline, or you will receive penalty.
- At least 5 files should be included:
  - ✦ `bidding_system.c`
  - ✦ `bidding_system_EDF.c`
  - ✦ `customer.c`
  - ✦ `Makefile`(as well as other `*.c` files)
  - ✦ `readme.txt`



# Punishment

- You will get **NO** credits if plagiarism.
- Late submission
  - **5%** for each day
- Error format
  - wrong file name/format
  - wrong output format



# Programming assignment 3

Due: 23:59 Tue, Dec 22, 2015

*–Enjoy your work :)*