[2015 Network System Programming Homework 8]

Upload:

1. Please compress your homework into **zip** or **tar** archive.
2. Naming rules: "**StudentID_SP_HW8.zip**".
   For example: M033040001_SP_HW8.zip
3. Upload your homework to **National Sun Yat-sen Cyber University.**
4. **Deadline: 2015/12/29 (Tue.) 23:59**

額外上傳規定：

1. 本次作業期限為一個月，但每個禮拜必須在**禮拜二晚上(23:59)前**上傳當週**進度報告及程式碼**至網路大學。
2. 進度報告須說明當週完成那些部分。
3. 進度報告格式不限。
4. 若有一週未繳交，則本作業**扣總分 10 分**。
5. 若提早完成，還是必須上傳。
6. **最後一週繳交完成的程式即可**。

Rules:

1. Please use **C language** in this homework and run your program on **Ubuntu 14.04**.
2. Please provide **Makefile** to compile your homework; otherwise, you will get **ZERO.**
3. **Do not copy homework of others (classmates, senior etc)**. If it happened, you will get **ZERO** whether you are either the owner of the homework or the copycat.
4. You have to deeply understand what your program do because TA will ask you something about your program during the demo.
5. If you have any question, please send email to unix_ta@net.nsysu.edu.tw or come to EC5018, but TA does not help to debug.
6. If you do not submit your assignment on time, you will not hand in the delayed homework and get **ZERO** as well. If you have trouble, please advise it in advance by email. Moreover, time and place for demo will be announced later.

◆ Motivation:

Implement a multi-thread client-server project.

◆ Description:

1. Here is the specification of a multi-thread client-server project.

2. General scenario:

   Client connect, sends message. Message is put into a file with the name of the destination.

   (1) Client-side.

   Client specifies a destination and a message. No code required - just use telnet. We'll have trust the clients to do it right, i.e. line1 is the destination, subsequent line the message.

   (2) Server-side.

   The server screen has a menu:

   "1) Display number of current connections"

   "2) Display statistics (average connect time etc.)"

   "3) Re-start statistics gathering"

   "4) Kill stale clients"

3. Implementation:

   Server consists of a number of separate threads:

   (1) A port-listener: sits in a forever accept loop and fires up a servlet thread for each new client.

   (2) A tidier and stats gatherer. This thread monitors the activity of all of the servlet threads. It also shuffles the array of data-structures representing these servlets. It commits completed messages to files, one per destination.

   (3) Servlet threads (one per client).

4. Issues. Some mutex locking and semaphores will be needed to ensure communication between the various threads.

5. Pthreads: you may need to use the following:

   (1) pthread_attr_init();

   (2) pthread_attr_setdetachstate();

   (3) pthread_cancel();

   (4) pthread_create();

   (5) pthread_mutex_lock();

(6) pthread_mutex_unlock();
(7) sem_wait();
(8) sem_post();
Consult the manual for details.

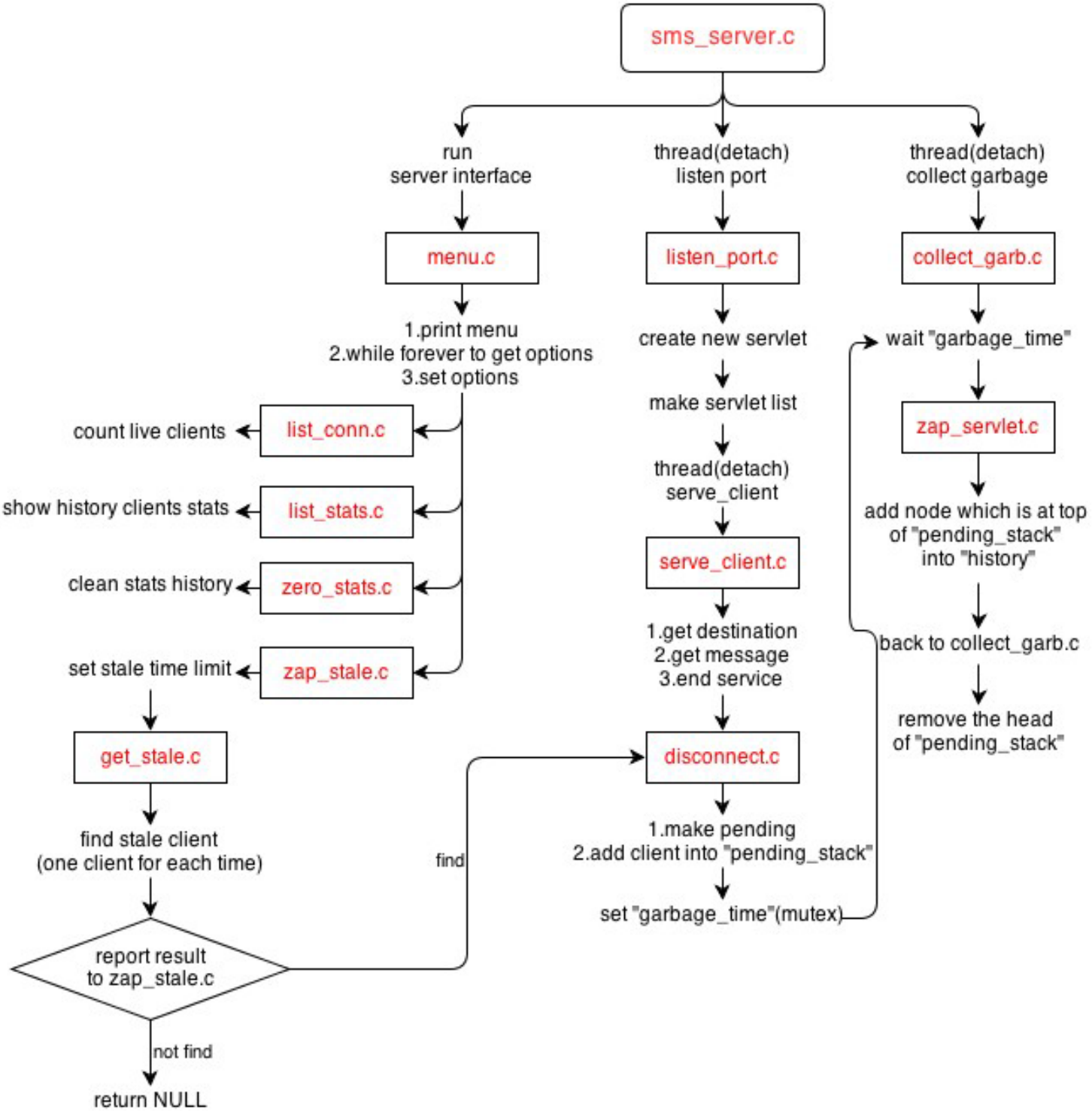
6. Files provided:

collect_garb.c	disconnect.c	get_stale.c	list_conn.c	list_stats.c
listen_port.c	menu.c	serve_client.c	sms_server.c	zap_servlet.c
zap_stale.c	zero_stats.c	sms.h	FuncSpec	Makefile

7. Data structures:
   (1) Servlet (door) is joined in a doubly linked list. It stores the current clients' information.
   (2) Stats (history) stores the offline client's information whether the client left correctly. It is a singly linked list and tend to be treated like stacks.
   (3) When a client left or was aborted, its servlet date will be removed from "Servlet list" and add its data into pending (pending_stack) waiting for doing the rest things and getting into "history". Pending (pending_stack) is also a singly linked list and tend to be treated like stacks.
   (4) Menu structure almost explains itself.

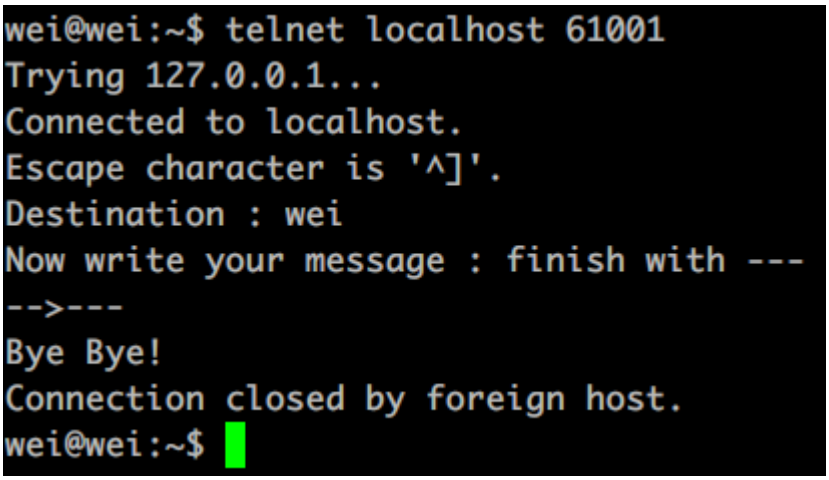
8. Port number:
   In sms.h, the rules of port number are as follows.
   (1) Freshman: 51 + student ID last 3 numbers
   (2) Sophomore: 52 + student ID last 3 numbers
   (3) Junior: 53 + student ID last 3 numbers
   (4) Seniors: 54 + student ID last 3 numbers
   (5) Master Degree 1st year: 61 + student ID last 3 numbers
   (6) Master Degree 2nd year: 62 + student ID last 3 numbers

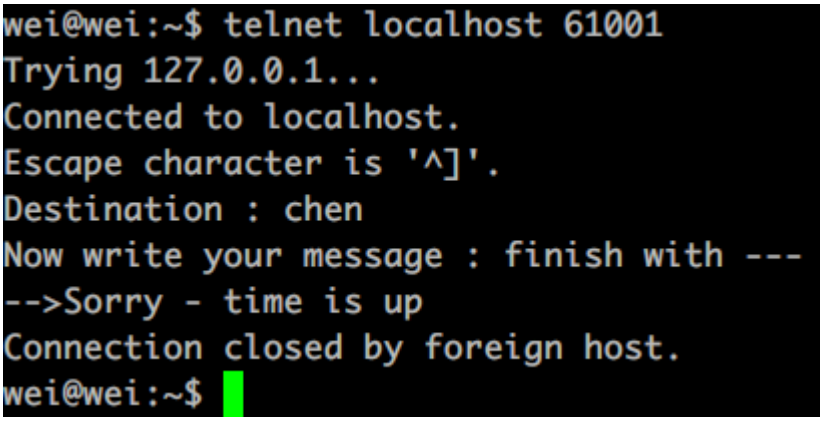
9. Flow chart:


sms_server.c
run
server interface
thread(detach)
listen port
thread(detach)
collect garbage
menu.c
listen_port.c
collect_garb.c
1.print menu
2.while forever to get options
3.set options
create new servlet
wait "garbage_time"
count live clients
list_conn.c
make servlet list
zap_servlet.c
show history clients stats
list_stats.c
thread(detach)
serve_client
add node which is at top
of "pending_stack"
into "history"
clean stats history
zero_stats.c
serve_client.c
set stale time limit
zap_stale.c
1.get destination
2.get message
3.end service
back to collect_garb.c
remove the head
of "pending_stack"
get_stale.c
disconnect.c
find stale client
(one client for each time)
find
1.make pending
2.add client into "pending_stack"
report result
to zap_stale.c
set "garbage_time"(mutex)
not find
return NULL

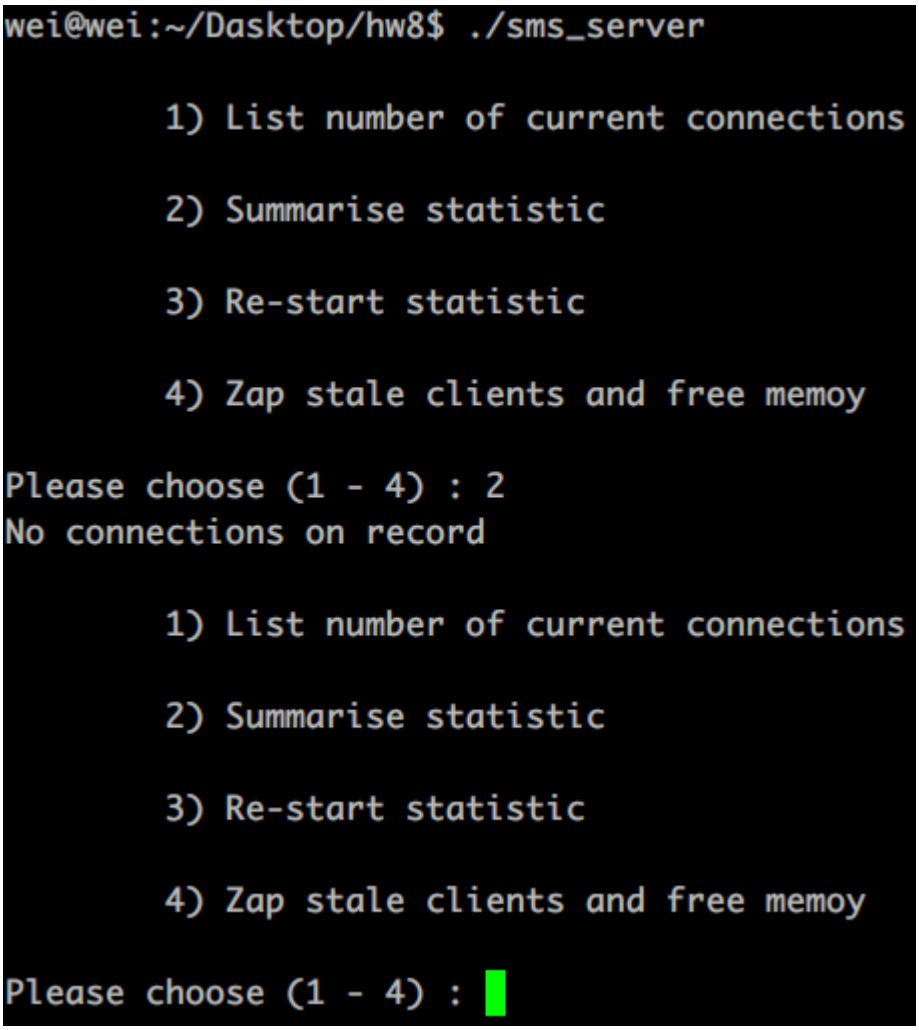
10. Sample output:

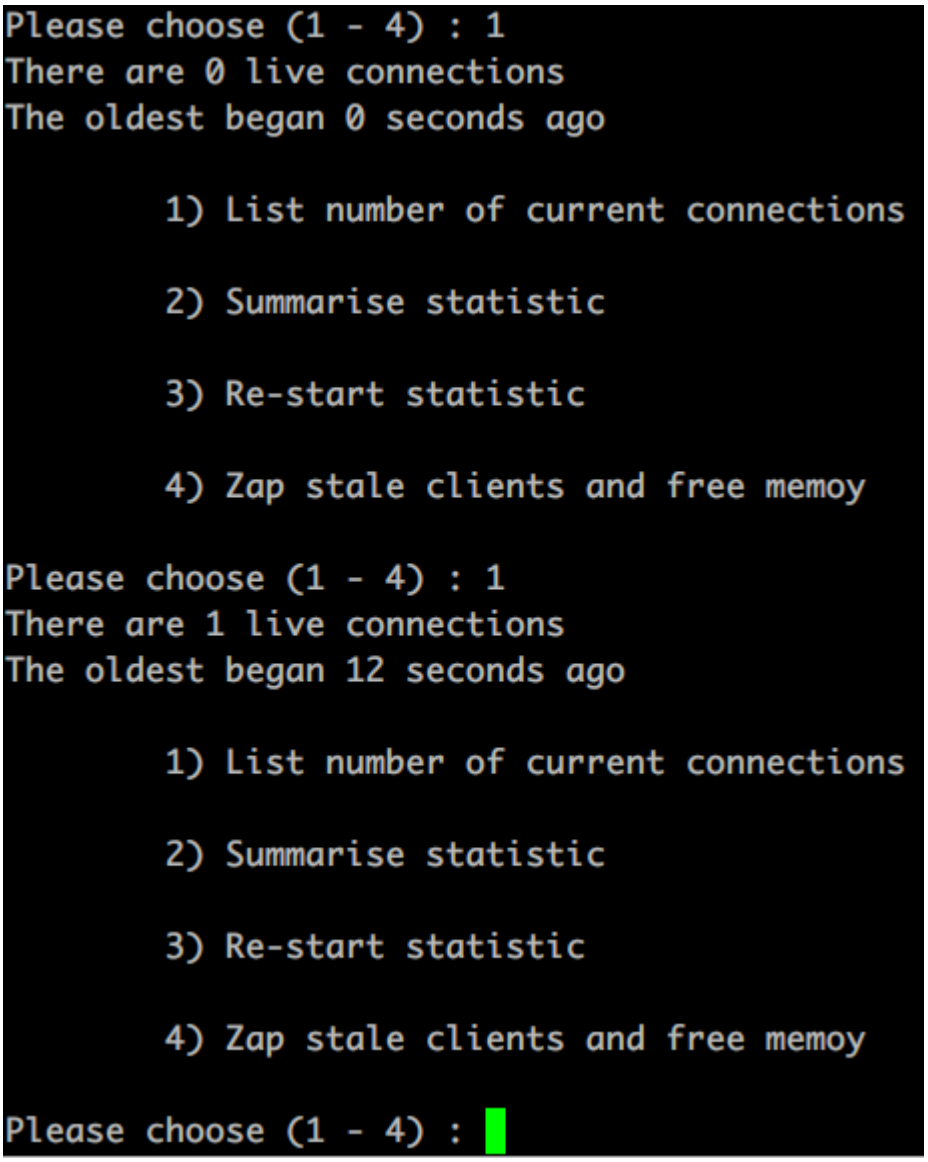
- Client:





- Server:


wei@wei:~/Dasktop/hw8$ ./sms_server
1) List number of current connections
2) Summarise statistic
3) Re-start statistic
4) Zap stale clients and free memoy
Please choose (1 - 4) : 2
No connections on record
1) List number of current connections
2) Summarise statistic
3) Re-start statistic
4) Zap stale clients and free memoy
Please choose (1 - 4) :


Please choose (1 - 4) : 1
There are 0 live connections
The oldest began 0 seconds ago
1) List number of current connections
2) Summarise statistic
3) Re-start statistic
4) Zap stale clients and free memoy
Please choose (1 - 4) : 1
There are 1 live connections
The oldest began 12 seconds ago
1) List number of current connections
2) Summarise statistic
3) Re-start statistic
4) Zap stale clients and free memoy
Please choose (1 - 4) :

```
Please choose (1 - 4) : 2
2 connections
0.0% aborted
average connect = 32.0 seconds
average size = 2.0 bytes
        1) List number of current connections

        2) Summarise statistic

        3) Re-start statistic

        4) Zap stale clients and free memoy

Please choose (1 - 4) :
```

```
Please choose (1 - 4) : 1
There are 1 live connections
The oldest began 64 seconds ago

        1) List number of current connections

        2) Summarise statistic

        3) Re-start statistic

        4) Zap stale clients and free memoy

Please choose (1 - 4) : 4
How many seconds counts as stale ? 60
Found a stale one

        1) List number of current connections

        2) Summarise statistic

        3) Re-start statistic

        4) Zap stale clients and free memoy

Please choose (1 - 4) :
```

```
Please choose (1 - 4) : 2
3 connections
33.3% aborted
average connect = 45.3 seconds
average size = 1.3 bytes
        1) List number of current connections

        2) Summarise statistic

        3) Re-start statistic

        4) Zap stale clients and free memoy

Please choose (1 - 4) :
```

```
Please choose (1 - 4) : 3

        1) List number of current connections

        2) Summarise statistic

        3) Re-start statistic

        4) Zap stale clients and free memoy

Please choose (1 - 4) : 2
No connections on record

        1) List number of current connections

        2) Summarise statistic

        3) Re-start statistic

        4) Zap stale clients and free memoy

Please choose (1 - 4) :
```