시스탬프로그래밍실습

#Assignment4-1

이름: 이진수

학번 : 2015722013

학과: 컴퓨터공학과

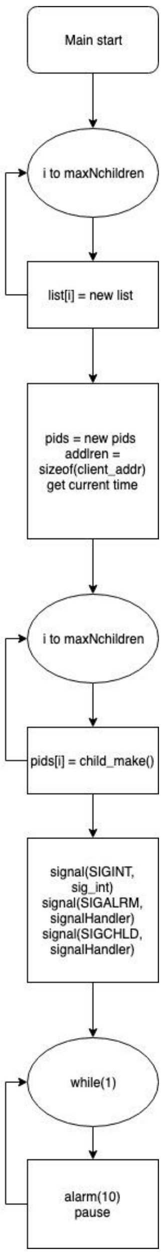
신영주 교수님

1. Introduction

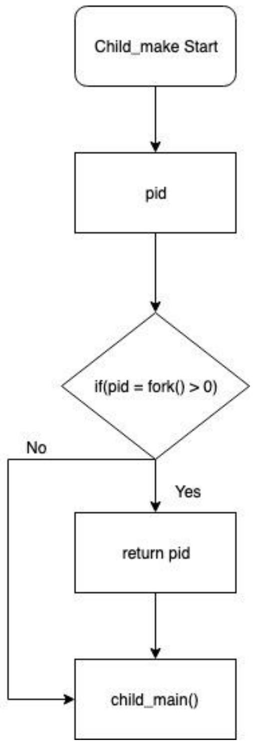
이번 과제는 저번 과제의 연장 선으로써 미리 5개의 자식 프로제서를 만들어 놓은 상태에서 클라이언트의 request를 받는 것이다. 저번에는 request를 받을 때 마다 fork 즉 자식 프로세스를 생성하여 request 처리를 해주었지만 이번에는 이미 있는 다섯개의 자식 프로세스에서 처리해 준다. 자식을 종료 시킬 때는 자식이 아닌 부모에게 signal을 줘서 프로세서에서 자식들을 죽이는 형태를 띈다.

2. Flow chart

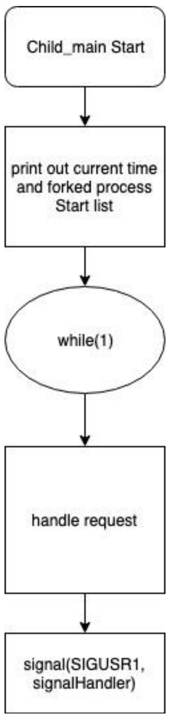
-Main function



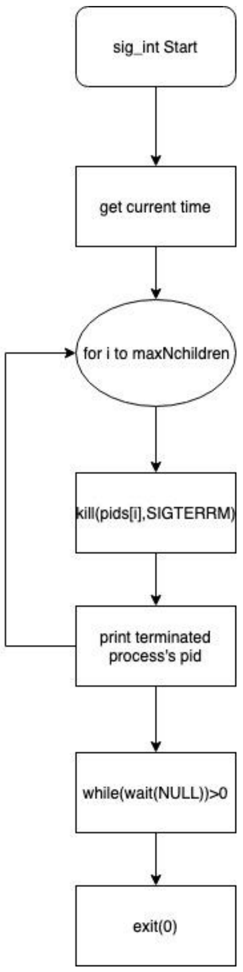
- Flow chart for Child make function



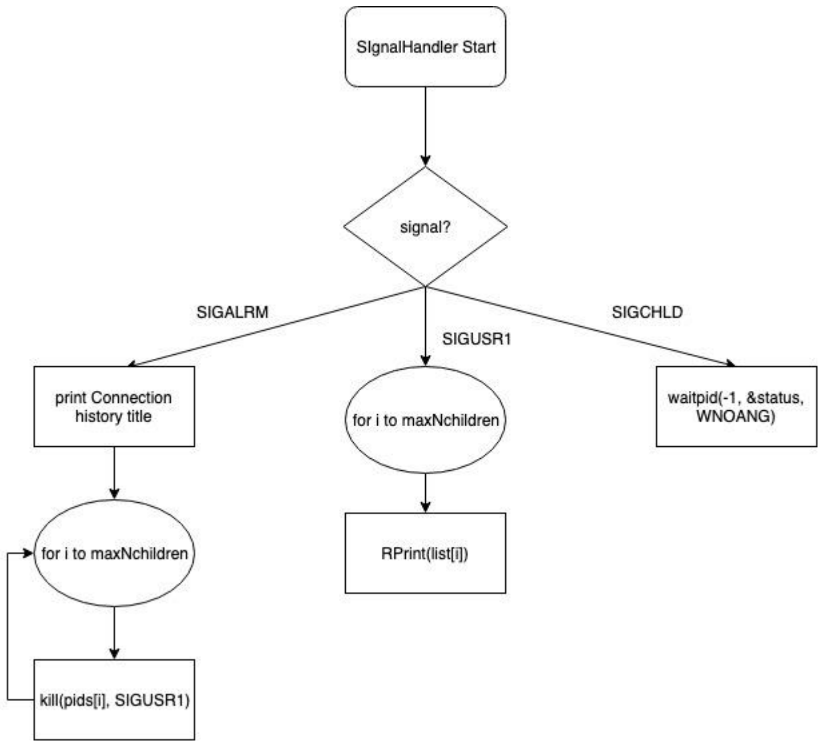
- Flow chart for Child main function


Child_main Start
print out current time and forked process Start list
while(1)
handle request
signal(SIGUSR1, signalHandler)

- Flow chart for sig_int function



- Flow chart for signalHandler



3. Pseudo code

- Pseudo code for main

```
main(){
        maxNchildren = 5;
        for(i 0 to maxNchildren){
                list[i] = new list;
        }

        setsocket(-----);

        pids = new pid
        addlren = sizeof(client_addr);

        getcurrenttime();

        for(i 0 to maxNchilderen){
                pids[i] = child_make(i, socket_fd, addrlen, client_addr, &list[i]);
        }

        signal(SIGINT, sig_int);
        signal(SIGALRM, signalHandler);
        signal(SIGCHLD, signalHandler);

        for(;;){
                alarm(10);
                pause;
        }
        close(soccket_fd)
}
```

- Pseudo code for child make

```
 pid_t child_make(i, socket_fd, addlren, client_addr, lsit, pid){
        pid_t pid;

        if((pid = fork())>0){
                return(pid);
        }

        child_main(i, socket_fd, addrlen, client_addr, list, pid);
}
```

- Pseudo code for child main

```
void child_main(i, socket_fd, addlren , client_addr, list, pid){
 get current time();
 print(getpid() process is forked);

 Start list;

 while(1){
        print(connected new client);
        Handle Rquest();
        print(disconnected client);
        signal(SIGUSR1, signalHandler);
 }
}
```

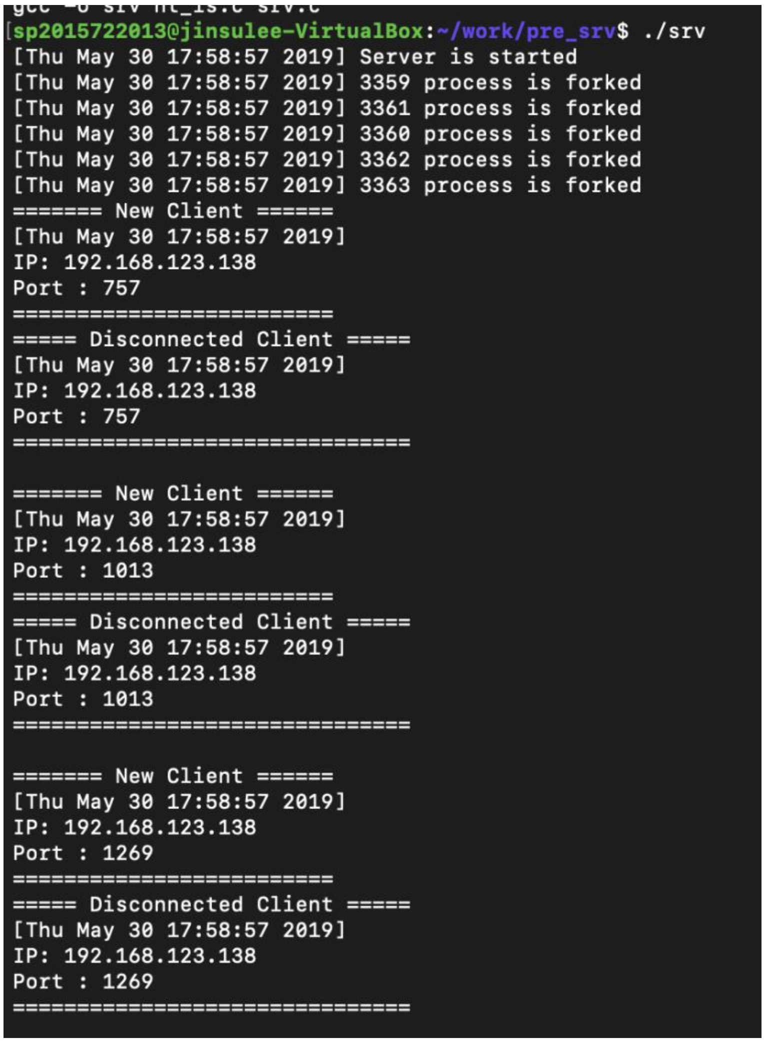
- Pseudo code for sig_int

```
void sig_int(signal){
 get current time();

 for(i 0 to maxNchildren){
        kill(pids[i], SIGTERM);
        print(which process is terminated);
 }
 print(Server is terminated);

 while(wati(NULL)>0)
 exit(0);
}
```

- Pseudo code for signalHandler

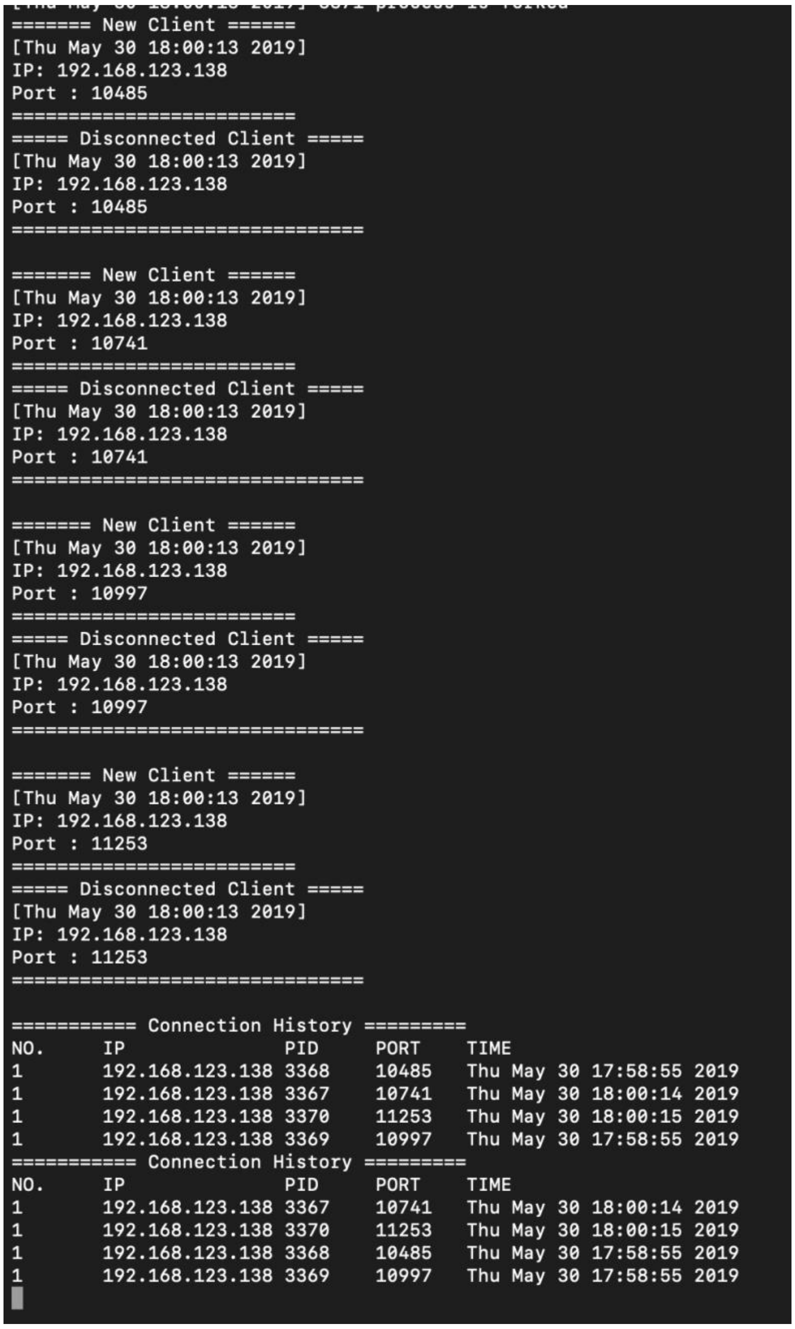
```
 void signalHandler(signal){
        if(signal == SIGALRM){
               print(connection history title);
               for(i 0 to maxNchildren){
                      kill(pids[i], SIGUSR1);
               }
        }

        if(signal == SIGUSR1){
               for(i 0 to maxNchildren){
                      RPrint(list[i]);
               }
        }

        if(signal == SIGCHLD){
               waitpid(-1, &status, WNOHANG);
        }
}
```

4. Result



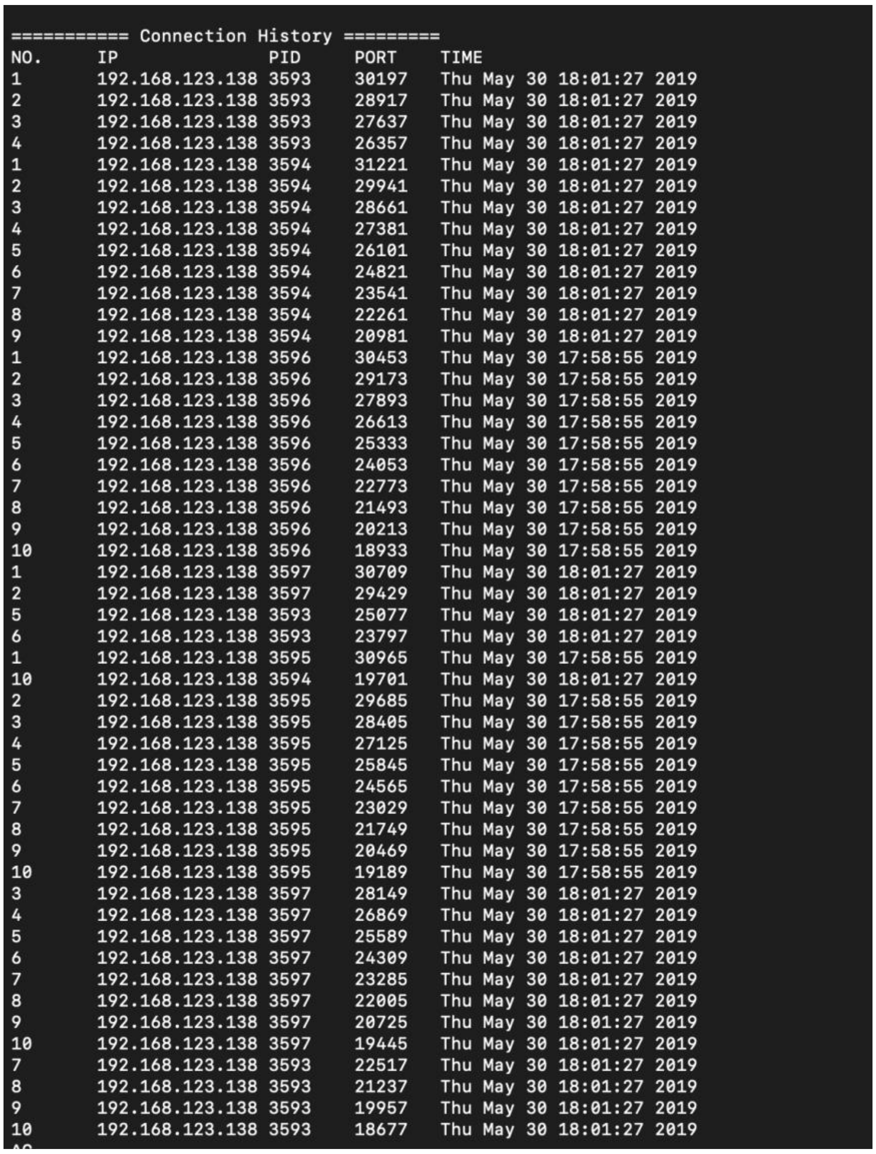
```
gcc -o srv ht_ls.c srv.c
[sp2015722013@jinsulee-VirtualBox:~/work/pre_srv$ ./srv
[Thu May 30 17:58:57 2019] Server is started
[Thu May 30 17:58:57 2019] 3359 process is forked
[Thu May 30 17:58:57 2019] 3361 process is forked
[Thu May 30 17:58:57 2019] 3360 process is forked
[Thu May 30 17:58:57 2019] 3362 process is forked
[Thu May 30 17:58:57 2019] 3363 process is forked
======= New Client ======
[Thu May 30 17:58:57 2019]
IP: 192.168.123.138
Port : 757
=========================
===== Disconnected Client =====
[Thu May 30 17:58:57 2019]
IP: 192.168.123.138
Port : 757
===============================

======= New Client ======
[Thu May 30 17:58:57 2019]
IP: 192.168.123.138
Port : 1013
=========================
===== Disconnected Client =====
[Thu May 30 17:58:57 2019]
IP: 192.168.123.138
Port : 1013
===============================

======= New Client ======
[Thu May 30 17:58:57 2019]
IP: 192.168.123.138
Port : 1269
=========================
===== Disconnected Client =====
[Thu May 30 17:58:57 2019]
IP: 192.168.123.138
Port : 1269
===============================
```

<result scene of initial state>



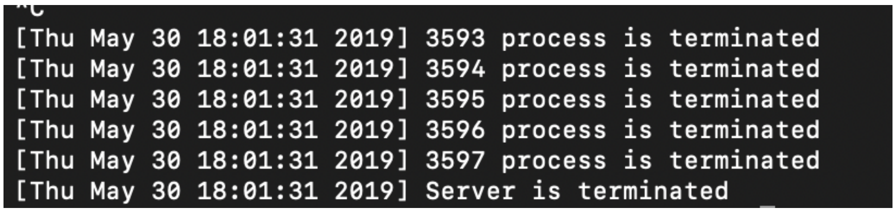
```
======= New Client ======
[Thu May 30 18:00:13 2019]
IP: 192.168.123.138
Port : 10485
=========================
===== Disconnected Client =====
[Thu May 30 18:00:13 2019]
IP: 192.168.123.138
Port : 10485
===============================

======= New Client ======
[Thu May 30 18:00:13 2019]
IP: 192.168.123.138
Port : 10741
=========================
===== Disconnected Client =====
[Thu May 30 18:00:13 2019]
IP: 192.168.123.138
Port : 10741
===============================

======= New Client ======
[Thu May 30 18:00:13 2019]
IP: 192.168.123.138
Port : 10997
=========================
===== Disconnected Client =====
[Thu May 30 18:00:13 2019]
IP: 192.168.123.138
Port : 10997
===============================

======= New Client ======
[Thu May 30 18:00:13 2019]
IP: 192.168.123.138
Port : 11253
=========================
===== Disconnected Client =====
[Thu May 30 18:00:13 2019]
IP: 192.168.123.138
Port : 11253
===============================

=========== Connection History =========
NO.     IP              PID     PORT    TIME
1       192.168.123.138 3368    10485   Thu May 30 17:58:55 2019
1       192.168.123.138 3367    10741   Thu May 30 18:00:14 2019
1       192.168.123.138 3370    11253   Thu May 30 18:00:15 2019
1       192.168.123.138 3369    10997   Thu May 30 17:58:55 2019
=========== Connection History =========
NO.     IP              PID     PORT    TIME
1       192.168.123.138 3367    10741   Thu May 30 18:00:14 2019
1       192.168.123.138 3370    11253   Thu May 30 18:00:15 2019
1       192.168.123.138 3368    10485   Thu May 30 17:58:55 2019
1       192.168.123.138 3369    10997   Thu May 30 17:58:55 2019
```

<result scene of printing Connection history>



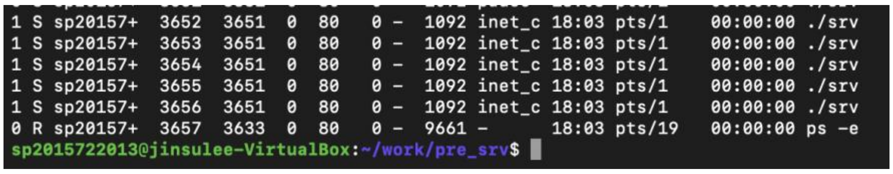
```
=========== Connection History =========
NO.     IP              PID     PORT    TIME
1       192.168.123.138 3593    30197   Thu May 30 18:01:27 2019
2       192.168.123.138 3593    28917   Thu May 30 18:01:27 2019
3       192.168.123.138 3593    27637   Thu May 30 18:01:27 2019
4       192.168.123.138 3593    26357   Thu May 30 18:01:27 2019
1       192.168.123.138 3594    31221   Thu May 30 18:01:27 2019
2       192.168.123.138 3594    29941   Thu May 30 18:01:27 2019
3       192.168.123.138 3594    28661   Thu May 30 18:01:27 2019
4       192.168.123.138 3594    27381   Thu May 30 18:01:27 2019
5       192.168.123.138 3594    26101   Thu May 30 18:01:27 2019
6       192.168.123.138 3594    24821   Thu May 30 18:01:27 2019
7       192.168.123.138 3594    23541   Thu May 30 18:01:27 2019
8       192.168.123.138 3594    22261   Thu May 30 18:01:27 2019
9       192.168.123.138 3594    20981   Thu May 30 18:01:27 2019
1       192.168.123.138 3596    30453   Thu May 30 17:58:55 2019
2       192.168.123.138 3596    29173   Thu May 30 17:58:55 2019
3       192.168.123.138 3596    27893   Thu May 30 17:58:55 2019
4       192.168.123.138 3596    26613   Thu May 30 17:58:55 2019
5       192.168.123.138 3596    25333   Thu May 30 17:58:55 2019
6       192.168.123.138 3596    24053   Thu May 30 17:58:55 2019
7       192.168.123.138 3596    22773   Thu May 30 17:58:55 2019
8       192.168.123.138 3596    21493   Thu May 30 17:58:55 2019
9       192.168.123.138 3596    20213   Thu May 30 17:58:55 2019
10      192.168.123.138 3596    18933   Thu May 30 17:58:55 2019
1       192.168.123.138 3597    30709   Thu May 30 18:01:27 2019
2       192.168.123.138 3597    29429   Thu May 30 18:01:27 2019
5       192.168.123.138 3593    25077   Thu May 30 18:01:27 2019
6       192.168.123.138 3593    23797   Thu May 30 18:01:27 2019
1       192.168.123.138 3595    30965   Thu May 30 17:58:55 2019
10      192.168.123.138 3594    19701   Thu May 30 18:01:27 2019
2       192.168.123.138 3595    29685   Thu May 30 17:58:55 2019
3       192.168.123.138 3595    28405   Thu May 30 17:58:55 2019
4       192.168.123.138 3595    27125   Thu May 30 17:58:55 2019
5       192.168.123.138 3595    25845   Thu May 30 17:58:55 2019
6       192.168.123.138 3595    24565   Thu May 30 17:58:55 2019
7       192.168.123.138 3595    23029   Thu May 30 17:58:55 2019
8       192.168.123.138 3595    21749   Thu May 30 17:58:55 2019
9       192.168.123.138 3595    20469   Thu May 30 17:58:55 2019
10      192.168.123.138 3595    19189   Thu May 30 17:58:55 2019
3       192.168.123.138 3597    28149   Thu May 30 18:01:27 2019
4       192.168.123.138 3597    26869   Thu May 30 18:01:27 2019
5       192.168.123.138 3597    25589   Thu May 30 18:01:27 2019
6       192.168.123.138 3597    24309   Thu May 30 18:01:27 2019
7       192.168.123.138 3597    23285   Thu May 30 18:01:27 2019
8       192.168.123.138 3597    22005   Thu May 30 18:01:27 2019
9       192.168.123.138 3597    20725   Thu May 30 18:01:27 2019
10      192.168.123.138 3597    19445   Thu May 30 18:01:27 2019
7       192.168.123.138 3593    22517   Thu May 30 18:01:27 2019
8       192.168.123.138 3593    21237   Thu May 30 18:01:27 2019
9       192.168.123.138 3593    19957   Thu May 30 18:01:27 2019
10      192.168.123.138 3593    18677   Thu May 30 18:01:27 2019
```

<connection history only shows up to 10 connection>



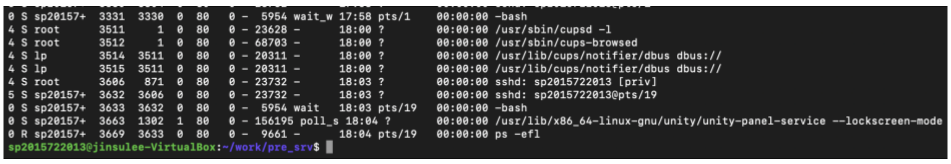
```
^C
[Thu May 30 18:01:31 2019] 3593 process is terminated
[Thu May 30 18:01:31 2019] 3594 process is terminated
[Thu May 30 18:01:31 2019] 3595 process is terminated
[Thu May 30 18:01:31 2019] 3596 process is terminated
[Thu May 30 18:01:31 2019] 3597 process is terminated
[Thu May 30 18:01:31 2019] Server is terminated
```

<result scene of end of connection>



```
1 S sp20157+  3652  3651  0  80   0 -  1092 inet_c 18:03 pts/1    00:00:00 ./srv
1 S sp20157+  3653  3651  0  80   0 -  1092 inet_c 18:03 pts/1    00:00:00 ./srv
1 S sp20157+  3654  3651  0  80   0 -  1092 inet_c 18:03 pts/1    00:00:00 ./srv
1 S sp20157+  3655  3651  0  80   0 -  1092 inet_c 18:03 pts/1    00:00:00 ./srv
1 S sp20157+  3656  3651  0  80   0 -  1092 inet_c 18:03 pts/1    00:00:00 ./srv
0 R sp20157+  3657  3633  0  80   0 -  9661 -      18:03 pts/19   00:00:00 ps -e
sp2015722013@jinsulee-VirtualBox:~/work/pre_srv$
```

<when maintaining connection>



```
0 S sp20157+  3331  3330  0  80   0 -  5954 wait_w 17:58 pts/1    00:00:00 -bash
4 S root      3511     1  0  80   0 - 23628 -      18:00 ?        00:00:00 /usr/sbin/cupsd -l
4 S root      3512     1  0  80   0 - 68703 -      18:00 ?        00:00:00 /usr/sbin/cups-browsed
4 S lp        3514  3511  0  80   0 - 20311 -      18:00 ?        00:00:00 /usr/lib/cups/notifier/dbus dbus://
4 S lp        3515  3511  0  80   0 - 20311 -      18:00 ?        00:00:00 /usr/lib/cups/notifier/dbus dbus://
4 S root      3606   871  0  80   0 - 23732 -      18:03 ?        00:00:00 sshd: sp2015722013 [priv]
5 S sp20157+  3632  3606  0  80   0 - 23732 -      18:03 ?        00:00:00 sshd: sp2015722013@pts/19
0 S sp20157+  3633  3632  0  80   0 -  5954 wait   18:03 pts/19   00:00:00 -bash
0 S sp20157+  3663  1302  1  80   0 - 156195 poll_s 18:04 ?       00:00:00 /usr/lib/x86_64-linux-gnu/unity/unity-panel-service --lockscreen-mode
0 R sp20157+  3669  3633  0  80   0 -  9661 -      18:04 pts/19   00:00:00 ps -efl
sp2015722013@jinsulee-VirtualBox:~/work/pre_srv$
```

<there is no zombie processor after disconnection>

5. Conclusion

이번 과제는 저번 과제의 연장 선이라고 생각했다. 저번 과제는 client가 request를 줄 때마다 fork를 생성하고 바로 끊어줬는데 이번에는 미리 5개의 fork를 생성하여 request를 받을 때 미리 생성한 fork의 processor에 할당을 해준다. 이번 과제를 하면서 가장 고민을 많이 했던 부분은 signal handler였다. Child processor는 parent processor와 완전 별개의 processor여서 signal을 어떻게 줘야 할지에 대한 생각을 많이 하였다. 저번처럼 parent가 alarm signal을 받으면 child에서 alarm signal을 받지 못하기 때문에 따로 child에게 전달하는 부분에서 난항을 겪었다. 이는 SIGUSR1을 통해 해결하였는데 signal alarm이 parent로 전달이 되면 signal handler에서 모든 children에게 SIGUSR1를 전달하여 child processor에서 connection history를 출력하게 만들었다. 이 고민을 하면서 저번 과제에서 구체적으로 알지 못했던 signal handler에 대해 구체적으로 알 수 있게 되었다.