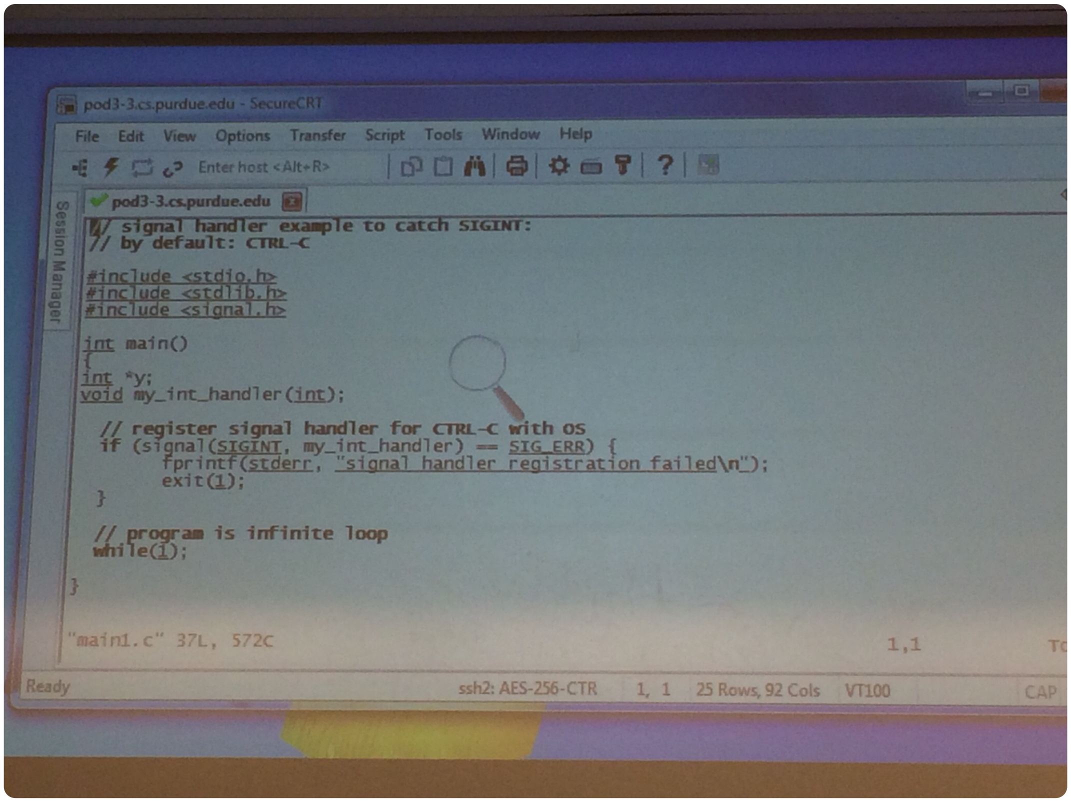
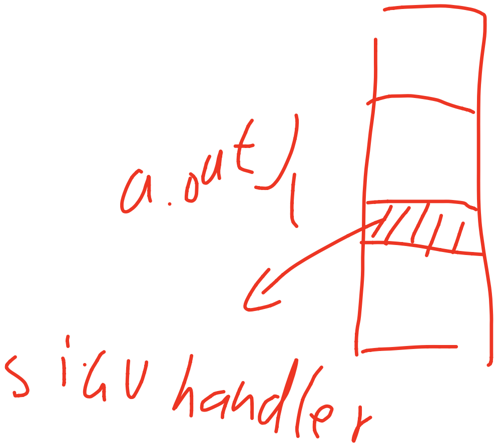
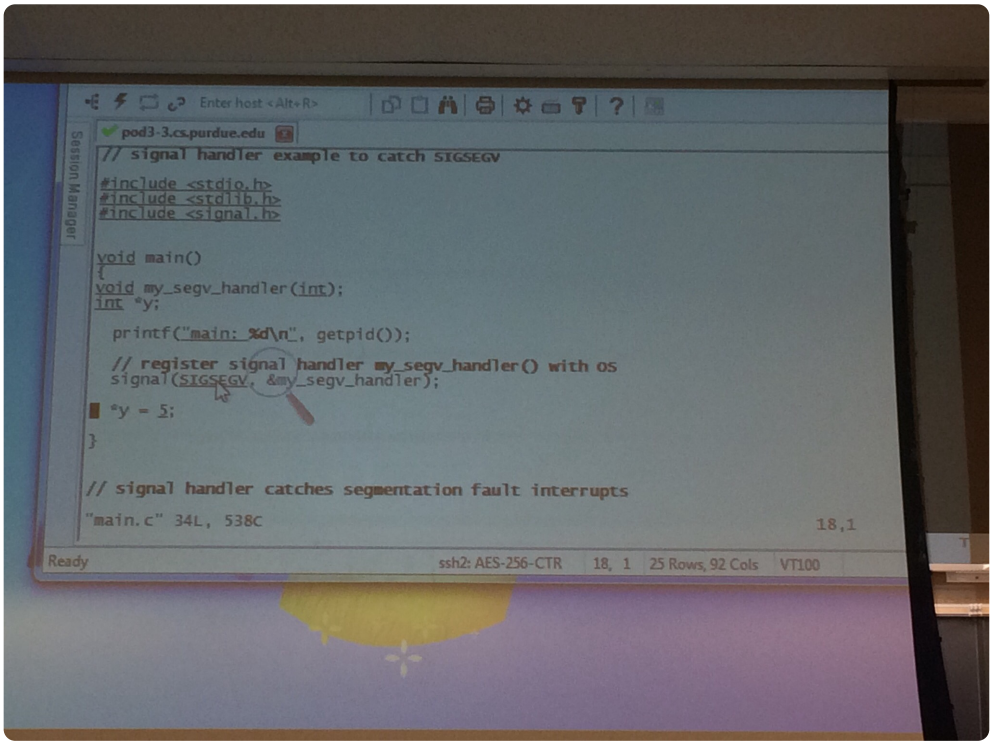


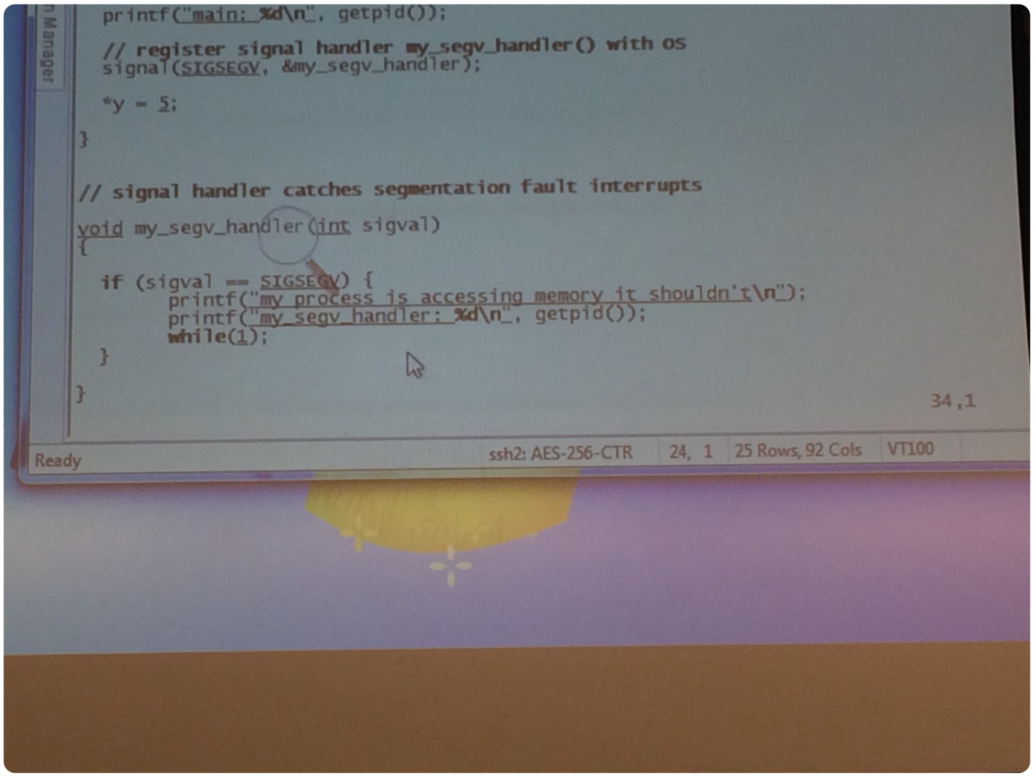
```
// signal handler example to catch SIGINT:
// by default: CTRL-C

#include <stdio.h>
#include <stdlib.h>
#include <signal.h>

int main()
{
int *y;
void my_int_handler(int);

  // register signal handler for CTRL-C with OS
  if (signal(SIGINT, my_int_handler) == SIG_ERR) {
        fprintf(stderr, "signal handler registration failed\n");
        exit(1);
  }

  // program is infinite loop
  while(1);
}
"main1.c" 37L, 572C                                                    1,1
```



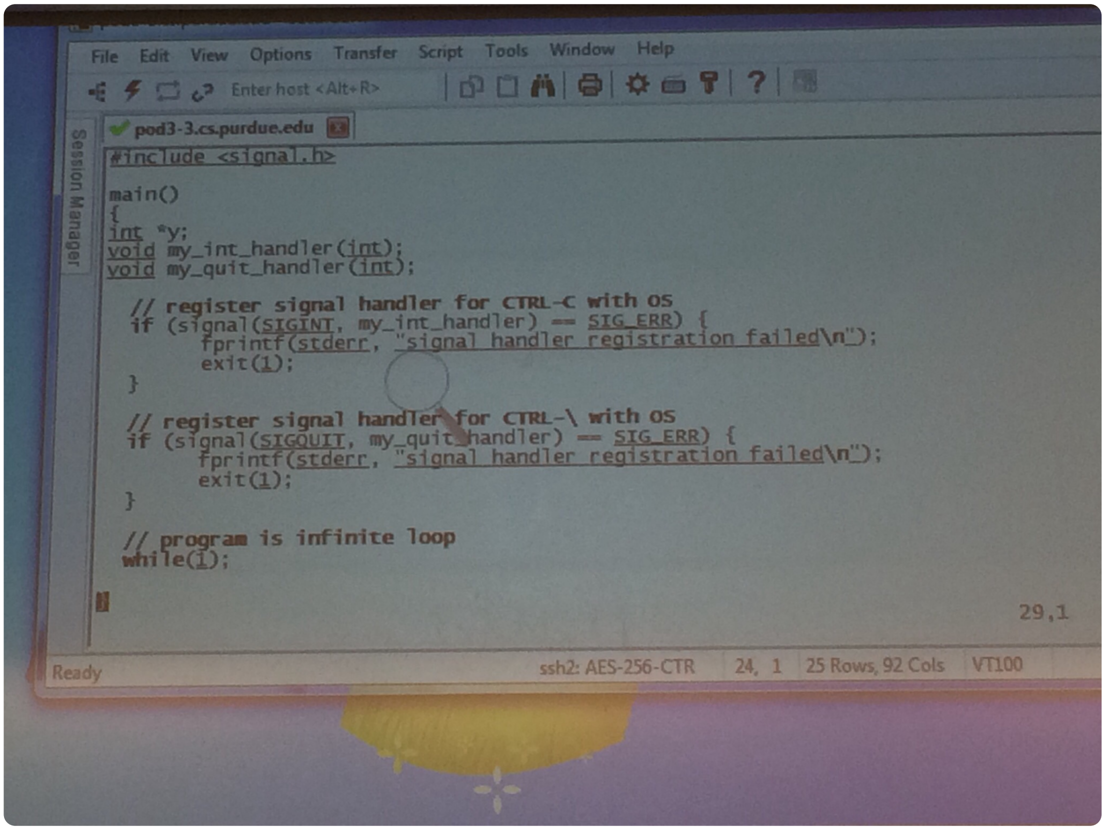
sigaction( ) to restore the default system signal handler.



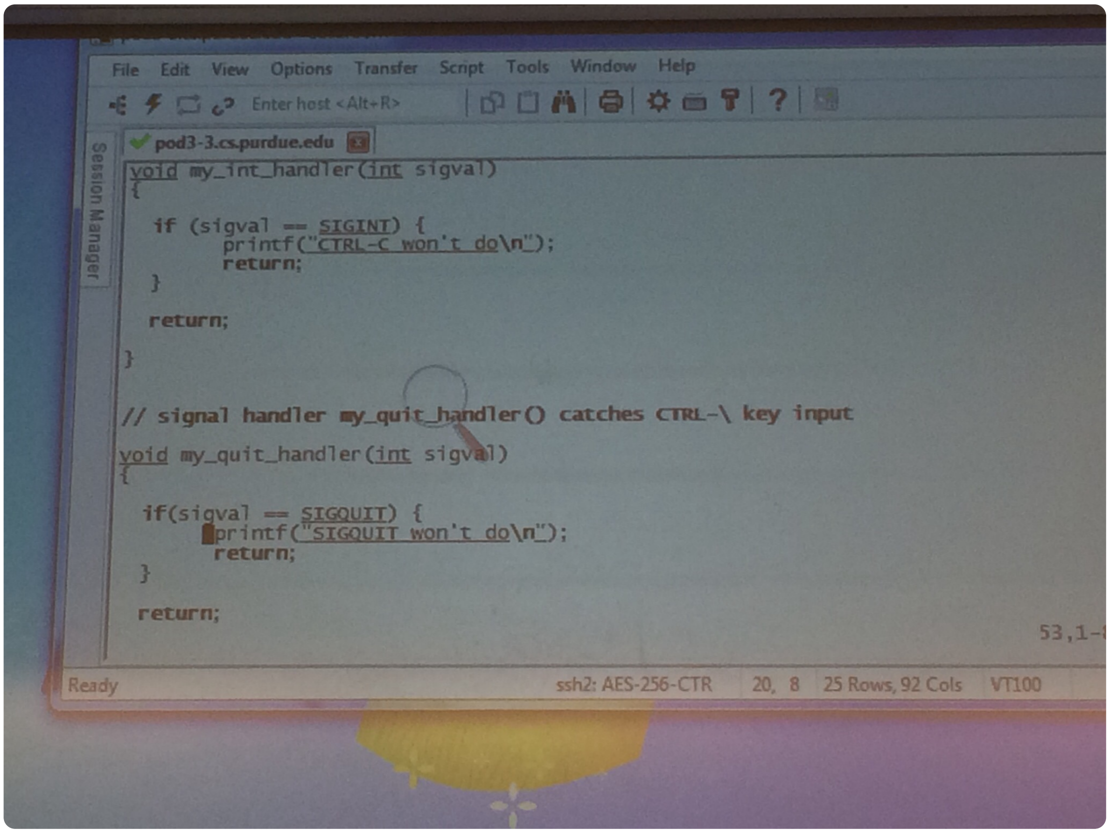
```
// signal handler example to catch SIGSEGV

#include <stdio.h>
#include <stdlib.h>
#include <signal.h>


void main()
{
void my_segv_handler(int);
int *y;

  printf("main: %d\n", getpid());

  // register signal handler my_segv_handler() with os
  signal(SIGSEGV, &my_segv_handler);

  *y = 5;

}


// signal handler catches segmentation fault interrupts
"main.c" 34L, 538C                                              18,1
```



```
  printf("main: %d\n", getpid());

  // register signal handler my_segv_handler() with os
  signal(SIGSEGV, &my_segv_handler);

  *y = 5;

}


// signal handler catches segmentation fault interrupts

void my_segv_handler(int sigval)
{

  if (sigval == SIGSEGV) {
        printf("my process is accessing memory it shouldn't\n");
        printf("my_segv_handler: %d\n", getpid());
        while(1);
  }

}
                                                                  34,1
```



```
#include <signal.h>

main()
{
int *y;
void my_int_handler(int);
void my_quit_handler(int);

  // register signal handler for CTRL-C with OS
  if (signal(SIGINT, my_int_handler) == SIG_ERR) {
        fprintf(stderr, "signal handler registration failed\n");
        exit(1);
  }

  // register signal handler for CTRL-\ with OS
  if (signal(SIGQUIT, my_quit_handler) == SIG_ERR) {
        fprintf(stderr, "signal handler registration failed\n");
        exit(1);
  }

  // program is infinite loop
  while(1);

                                                                29,1
```



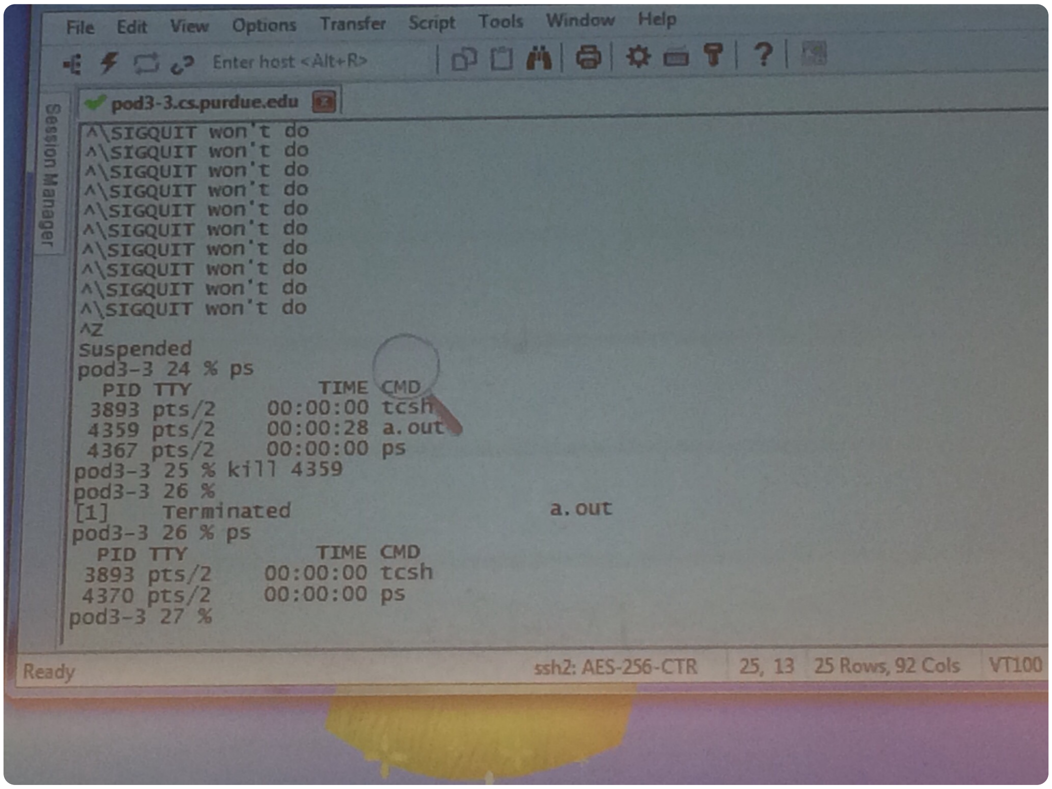
```
void my_int_handler(int sigval)
{

  if (sigval == SIGINT) {
        printf("CTRL-C won't do\n");
        return;
  }

  return;

}

// signal handler my_quit_handler() catches CTRL-\ key input

void my_quit_handler(int sigval)
{

  if(sigval == SIGQUIT) {
        printf("SIGQUIT won't do\n");
        return;
  }

  return;
                                                              53,1-
```

use ctrl-z to suspend process

use kill <pid> to kill the process

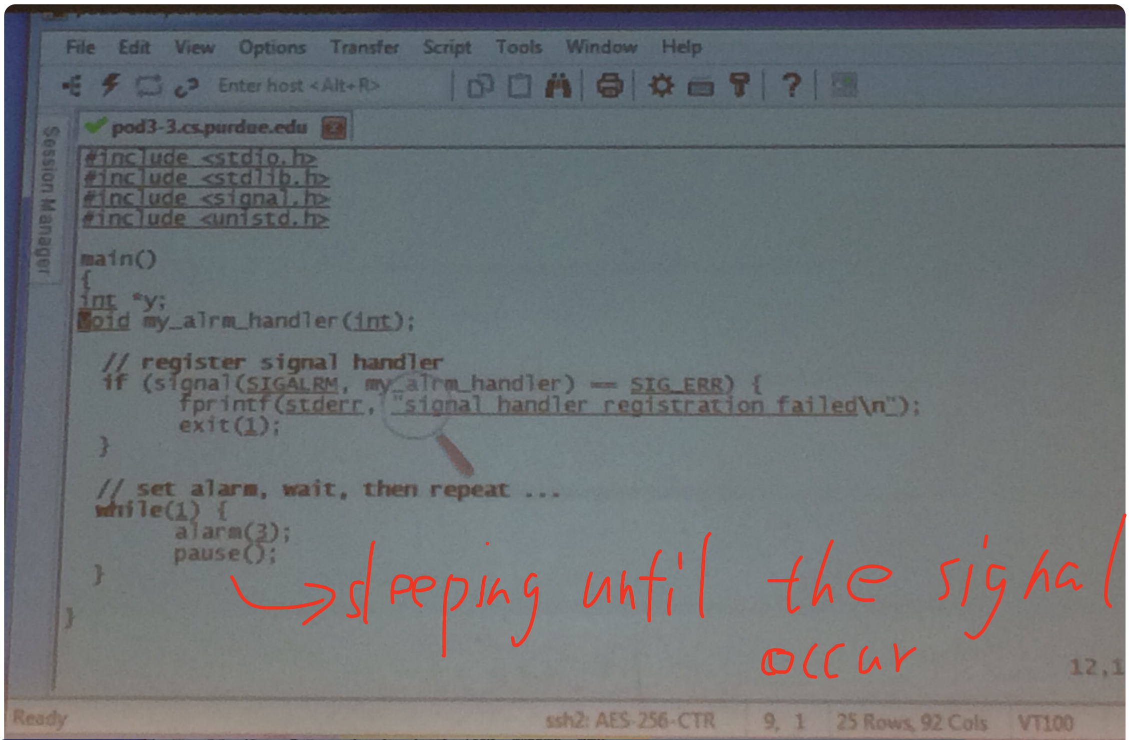
SIGSTOP , SIGKILL

two signal can't be catched



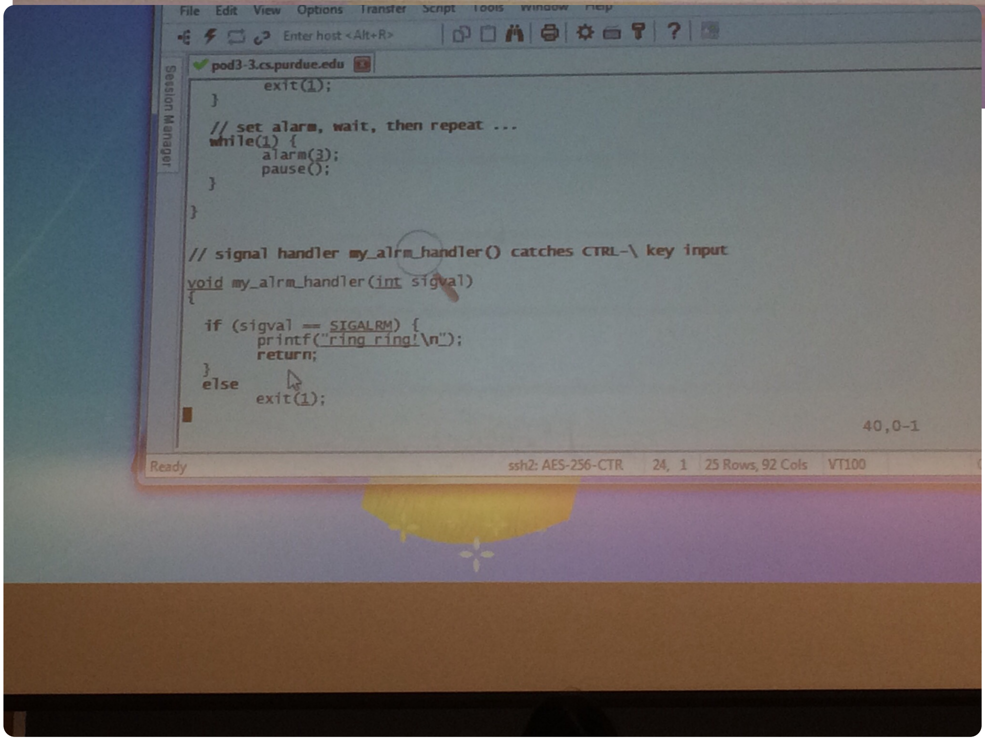
```
^\SIGQUIT won't do
^\SIGQUIT won't do
^\SIGQUIT won't do
^\SIGQUIT won't do
^\SIGQUIT won't do
^\SIGQUIT won't do
^\SIGQUIT won't do
^\SIGQUIT won't do
^\SIGQUIT won't do
^\SIGQUIT won't do
^Z
Suspended
pod3-3 24 % ps
  PID TTY          TIME CMD
 3893 pts/2    00:00:00 tcsh
 4359 pts/2    00:00:28 a.out
 4367 pts/2    00:00:00 ps
pod3-3 25 % kill 4359
pod3-3 26 %
[1]    Terminated                    a.out
pod3-3 26 % ps
  PID TTY          TIME CMD
 3893 pts/2    00:00:00 tcsh
 4370 pts/2    00:00:00 ps
pod3-3 27 %
```

main
SIGALRM



```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <unistd.h>

main()
{
int *y;
void my_alrm_handler(int);

  // register signal handler
  if (signal(SIGALRM, my_alrm_handler) == SIG_ERR) {
        fprintf(stderr, "signal handler registration failed\n");
        exit(1);
  }

  // set alarm, wait, then repeat ...
  while(1) {
        alarm(3);
        pause();
  }
}
                                                          12,1
Ready                         ssh2: AES-256-CTR   9, 1   25 Rows, 92 Cols   VT100
```

→ sleeping until the signal occur



```
        exit(1);
  }

  // set alarm, wait, then repeat ...
  while(1) {
        alarm(3);
        pause();
  }
}

// signal handler my_alrm_handler() catches CTRL-\ key input

void my_alrm_handler(int sigval)
{
  if (sigval == SIGALRM) {
        printf("ring ring!\n");
        return;
  }
  else
        exit(1);
                                                          40,0-1
Ready                         ssh2: AES-256-CTR   24, 1   25 Rows, 92 Cols   VT100
```

SIGIO , SIGPOLL

web client get notice of internet pacakage.