Homework 9

Problem 1

Modify *doit*:

```
void doit(int fd)
{
//...
15 if (strcasecmp(method, "GET") && strcasecmp(method, "POST")) {
//...
19 }
// ...
29 if (strcasecmp(method, "POST")) { /* Serve POST request */
    if (!(S_ISREG(sbuf.st_mode)) || !(S_IXUSR & sbuf.st_mode)) {
        clienterror(fd, filename, "403", "Forbidden",
        "Tiny couldn't run the CGI program");
        return;
    }
    serve_dynamic_post(fd, filename);
}
30 if (is_static) { /* Serve static content */
//...
}
```

Add a function *serve_dynamic_post*: (Mostly like serve_dynamic, need to redirect STDIN to client fd)

```
void serve_dynamic_post(int fd, char *filename)
{
char buf[MAXLINE], *emptylist[] = { NULL };

/* Return first part of HTTP response */
sprintf(buf, "HTTP/1.0 200 OK\r\n");
Rio_writen(fd, buf, strlen(buf));
sprintf(buf, "Server: Tiny Web Server\r\n");
Rio_writen(fd, buf, strlen(buf));

if (Fork() == 0) { /* child */
   Dup2(fd, STDOUT_FILENO); /* Redirect stdout to client */
   Dup2(fd, STDIN_FILENO); /* Redirect stdin to client */
   Execve(filename, emptylist, environ); /* Run CGI program */
}
Wait(NULL); /* Parent waits for and reaps child */
}
```

Problem 2

A. Because *select()* will set the ready_set. Program should modify it to read_set before each *select()* call.

B. Pros: 1. Programmers can control the execute order of the program.

2. Easy to share data between events & easy to debug.

3. Faster than processes/threads.

Cons: 1. Code is complex.

2. Programmers should maintain the granularity.

3. Cannot fully utilize multi-core processors.

Problem 3

A. File descriptor table.	thread
B. File table.	process
C. Stack.	not shared
D. Heap.	thread
E. Program counter.	not shared
F. Condition code.	not shared
G. Installed handler.	thread
H. V-node table.	process