CSE 376 Homework 4 Design Document

Kenny Liang

Adam Tringali

Harsh Vig

Table of Contents

Commands

Here are some sample commands of the client

1. list command will display all jobs along with their current information(pid,return value,status,etc)

   list

2. Stop,Suspend,or Continue a job command. The argument following the command name is the job number to stop,suspend,or continue on.

kill 1	**- stops job number 1**
susp 2	**- suspends job number 2**
cont 3	**- continues job number 3**
nice 1 -10	**- changes the priority of job number 1 to -10 (job # must come first)**

3. Adding a job command. argv= specifies the argument strings (program name is always the first argument) with ‘,’ as delimiter. envp= specifies the environment variable strings with ‘-’ as delimiter. nice= specifies the priority value to run the program. **argv is always required.**

   add argv=echo,hello,world envp=myVar-121,myVar-322 nice=-10
   add argv=ps,-le
   add argv=du,-sh,/usr nice=10

Client

Usage : ./client [-h] [-p pathname]

-h displays the usage string and the flag descriptions along with client commands descriptions
-p pathname - specifies the file of the domain socket to connect to

View the commands and protocol section of the design document for understanding client functionality.

Brief summary: client continuously looks to read from stdin for user commands then processes and sends the request to the server, looks to read from the pipe for any response of programs. **Program responses are stored in files so the client has to open the file, store the file content in a buffer and print it, then unlink the file. Reads from stdin and pipe are non-blocking.**

Protocol

Communication between server and client is done through sending “packets”. A packet is a constant number of bytes sent which contains information for a type of request or response between server and client. Packets allow us to know exactly how many bytes we need to read\write to the pipe communication. As a result, we won’t read\write too little or too much.

Here is the structure of a packet of our program. It contains 16 bytes (4 ints) to store the necessary information we need for a request or response. Type will contain a value that determines what type of request or response it is. **Field1,field2,field3 can contain arguments to request or responses like job number or priority value, or additional bytes to read. Additional bytes to read is required for requests like adding a job, which needs the argv,envp string. Field 1 will contain the length of a response string or file for the client to read from the file or pipe.**

```
typedef struct packet {
    int type;   //  type of request\response
    int field1; //  stores argc or job number
    int field2; //  stores envp
    int field3; //  stores the priority value
} PACKET;
```

Here is the types of packets that can be sent to our server or client

```
typedef enum {
    NONE_PKT,
    // client request pkts
    ADD_PKT,LISTREQ_PKT,KILL_PKT,SUSP_PKT,CONT_PKT,NICE_PKT,
    // server response pkts
    ERR_PKT,OUTPUT_PKT,LISTRES_PKT
}PACKET_TYPE;
```

Example steps of a client\server communication for adding a job

1. Client parses the add command from the user and retrieves the argv string and envp string, calculating the length of the argv string and envp string.
2. Client constructs a packet and assigns the type of pkt to ADD_PKT and field1 to be length of argv string and field2 to be length of envp string
3. Client then writes to the server the packet,then the argv string and then the envp string.
4. Server reads the size of a packet, checks the type of packet and sees it is an ADD_PKT. It then does additional read of pkt->field1 bytes and pkt->field2 bytes for the argv and envp strings. Server then continues to execute the job.

Server

Usage : ./server [-h] [-p pathname]

-h displays the usage string and the flag descriptions
-p pathname - specifies the file of the domain socket to create and listen on

The maximum concurrent number of jobs is 3, defined in job.h. The server can handle a maximum of 100 concurrent clients. Each client is given their own process to run their jobs therefore other clients will be unable to see the jobs of other clients. Each client process can then create processes to run jobs.

Server can execute programs called “jobs”. The server keeps track of all jobs for a client in a list. Server can return the information of all jobs into a file and send a response packet “OUTPUT_PKT” to the client with the filename containing the information of all jobs. The server can stop,suspend,continue, or change the priority of jobs.

```
typedef struct job{
    int jobNum;
    pid_t pid;
    char *argv;
    char *envp;
    char *status;
    struct job *next;
    int retVal;
    int priority;
    char *outfile;
    double usrtime;
    double systime;
}JOB;
```

Testing

View the examples.txt for examples of client commands and programs to run for testing functionality.