Problem Set 4 :The PIPE

Scott Jin

2017-11-15

Contents

List of Figures

1 Code Listings

Scott Jin——Wordgen

```
1  /*
2   ============================================================================
3   Name        : Wordgen.c
4   Author      : Zhekai Jin
5   Version     :
6   Copyright   : Your copyright notice
7   Description : Hello World in C, Ansi-style
8   ============================================================================
9   */
10
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <time.h>
14 #include <errno.h>
15 #include <string.h>
16 #define STR_LEN 10
17 #define CHAR_MIN 'A'
18 #define CHAR_MAX 'Z'
19
20 int wordgen(){
21   char str[STR_LEN + 1] = {0};  /*cstring formatting*/
22   int i;
23   int a = rand() % (STR_LEN-2)+3;     /*endless loop*/
24   for(i = 0; i < a; i ++)
25   {
26     str[i] = rand()%(CHAR_MAX-CHAR_MIN + 1) + CHAR_MIN;
27   }
28   printf("%.*s\n", a, str); /*keep c string format*/
29   return 0;
30 }
31 int main(int argc, char **argv) {
32   srand(time(NULL));
33     const char *nptr = argv[1];
34   long int count;
35
36   if(argc>2){
37     fprintf(stderr,"Too much arguments Provided, Only one optional argument needed\n");
38     exit(-1);
39   }else{
40     if(argc==1){
41       while(1){
42         wordgen();
43       }
44     }else{
45       errno=0;
46       count=strtol(argv[1],NULL,10);
47       if(nptr == NULL){
48         fprintf(stderr,"Please Enter an Valid number:%s->%s",argv[1],strerror(errno));
49         exit(-1);
50       }
51       if(errno){
52         fprintf(stderr,"Possible Overflow:%s,%s",argv[1],strerror(errno));
53         exit(-1);
54       }
55       for(int i=0;i<count;i++){
56         wordgen();
57       }
58     }
59   }
60   return EXIT_SUCCESS;
61 }
```

```c
/*
 * wordsearch.c
 * Created on: Nov 9, 2017
 * Author: scott   --> ADD Error Checking
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
#include <errno.h>
#include <ctype.h>
#define SIZE 1000000
int hashCode(char *str) {
      int hash = 0;
      for (int i = 0; i < strlen(str); i++) {
          if(hash>2147483647/32) return hash%SIZE; /* PLease Do not segfault*/
          hash = 31 * hash + str[i];
      }
      return hash % SIZE;
  }  /*hashcode*/
char* search(char* key,char** hashArray) {
   int hashIndex = hashCode(key);
   while(hashArray[hashIndex] != NULL) {
      if(!strcmp(hashArray[hashIndex],key))
         return hashArray[hashIndex];
      ++hashIndex;
      hashIndex %= SIZE;
   }
   return NULL;
}
void insert(char* key,char** hashArray) {
   int hashIndex = hashCode(key);
   while(hashArray[hashIndex] != NULL) {
      ++hashIndex;
      hashIndex %= SIZE;
   }
   hashArray[hashIndex] = key;
}
void display(char** hashArray) {
   int i = 0;
   for(i = 0; i<SIZE; i++) {
      if(hashArray[i] != NULL)
         printf("(%s)",hashArray[i]);
      else
         printf("");
   }
   printf("\n");
}

long int matched;

void signalhandler(){
  fprintf(stderr, "Mateched:%ld Words\n",matched);
  exit(13);
}
int main(int argc, char **argv) {
  long int accepted=0,rejected=0;
    ssize_t nread;
    size_t len = 0;
  FILE *fp;
    char *line = NULL;
    signal(SIGPIPE,signalhandler);
  char** hashArray = (char **)calloc(SIZE,sizeof(char*));
  if(hashArray==NULL){
    fprintf(stderr, "Error: Can't allocate memory for hashArray :%s\n",strerror(errno));
    exit(-1);
```

```
67    }
68    if(argc!=2) {
69      fprintf(stderr, "Error:␣Please␣specify␣the␣Dictionary␣file␣path\n");
70      exit(-1);
71    }
72    if ((fp=fopen(argv[1], "r")) == NULL) {
73      fprintf(stderr, "Error:␣Can't␣open␣input␣file␣%s:%s\n",argv[1],strerror(errno));
74      exit(-1);
75    }else{
76      while ((nread = getline(&line, &len, fp)) != -1) {
77        int i=0;
78        int isalpha=1;
79        while(line[i]){
80          if((line[i]>='a' && line[i]<='z') || (line[i]>='A' && line[i]<='Z') || (line[i] == '
                \n') || (line[i] == '\r') ){
81            if (line[i]>='a' && line[i]<='z') line[i]=toupper(line[i]);
82          }else{
83            isalpha=0;
84          }
85          i++;
86        }
87        if(isalpha){
88          insert(line,hashArray);
89          accepted++;
90        }else{
91          rejected++;
92        }
93      }
94    }
95    if (fclose (fp)) {
96        fprintf(stderr, "Error:␣Can't␣open␣input␣file␣%s:%s\n",argv[1],strerror(errno));
97        exit(-1);
98    }
99    fflush(stdout);
100   fprintf(stderr, "%ld␣words␣Accepted,␣%ld␣words␣Rejected\n",accepted,rejected);
101   while ((nread = getline(&line, &len, stdin)) != -1) {
102     if(search(line,hashArray)){
103       fputs(line,stdout);  /* echo the word*/
104       matched++;
105     }
106   }
107   fprintf(stderr, "Mateched:%ld␣Words\n",matched);
108   free(hashArray);
109   return 0;
110 }
```

Scott Jin——Pager

```
1  /*
2   * Pager.c
3   *
4   *  Created on: Nov 10, 2017
5   *      Author: scott
6   */
7  #include <fcntl.h>
8  #include <errno.h>
9  #include <unistd.h>
10 #include <stdio.h>
```

```c
#include <string.h>
#include <stdlib.h>
#define PAGE_SIZE 23
unsigned long long int count;
void ifexit(const char* line, char* buffer, int fd_in){
  if(!strcmp(line,"q") || !strcmp(line,"Q")){
    puts("***␣Pager␣terminated␣by␣Q␣command␣***");
    free(buffer);
    if(fd_in) close(fd_in); /*dont close 0 I guess*/
    exit(0);
  }
}
int main(int argc, char **argv) {
  int buffersize=125; int fd_in=0;int read_in=0;
  char *line = NULL;
  char *buffer = (char *)calloc(buffersize,sizeof(char*));
  if(buffer == NULL){
    fprintf(stderr,"No␣Memory␣available:%s\n",strerror(errno));
    exit(-1);
  }
  ssize_t nread;    size_t len = 0;
  int clag = 1;
  if((fd_in=open("/dev/tty",O_RDWR,S_IRUSR|S_IWUSR))==0){
    fprintf(stderr,"Can't␣open␣special␣file␣dev/tty:%s\n",strerror(errno));
    exit(-1);
  }
  while (1){
    while (clag) {
      nread = getline(&line, &len, stdin);
      if(nread!= -1){
        ifexit(line,buffer,fd_in);
        fputs(line,stdout);  /* echo the word*/
        count++;
        clag = count%(PAGE_SIZE+1);
      }else{
        if(errno) fprintf(stderr,"getline␣error:%s",strerror(errno));
        return 0;
      }
    }
    if(write(fd_in,"␣␣␣␣---Press␣RETURN␣for␣more---␣␣␣␣␣\n",30)==-1){
      fprintf(stderr,"Can't␣Write␣to␣special␣file␣dev/tty:%s\n",strerror(errno));
      exit(-1);
    }
    read_in = read(fd_in, buffer, buffersize);
    if(read_in == -1){
      fprintf(stderr,"Can't␣read␣from␣special␣file␣dev/tty:%s",strerror(errno));
      exit(-1);
    }
    if(!strcmp(buffer,"\r")||!strcmp(buffer,"\n")) clag=(++count)%(PAGE_SIZE+1);
    buffer[1] = 0;
    ifexit(buffer,buffer,fd_in);
  }
  return 0;
}
```

Scott Jin——Launcher

```c
/*
 * launcer.c
 *
 *  Created on: Nov 14, 2017
```

```
 *      Author: scott
 */
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/time.h>
#include <sys/resource.h>
#include <sys/types.h>

int IOredir(int fdold, int fdnew){
  if (dup2 (fdold, fdnew) < 0) {
    fprintf(stderr, "Warning:Error in dup2 target file discripter:=%d dup2() failure: %s\n",
         fdnew, strerror (errno));
      exit(EXIT_FAILURE);
  }
  if (close(fdold)<0){
    fprintf (stderr,"Warning:Error in closing file discripter(%d):%s[dangling file
        discripter exits]\n",fdold, strerror (errno));
    exit(EXIT_FAILURE);
  }
  return 0;
}
void CLOSE(int fdtodel){
  if (close(fdtodel)<0){
    fprintf (stderr, "Warning:Error in closing file discripter[%d]:%s[dangling file
        discripter exits]\n",fdtodel, strerror (errno));
    exit(EXIT_FAILURE);
  }
}
int main(int argc, char **argv){
  /*make fd for pipes*/
    int pipefd1[2];
    int pipefd2[2];
    int waitStatus;
  pid_t child;
  pid_t grandchild;
  pid_t greatgrandchild;
  if (argc >2) {
      fprintf(stderr, "Usage: %s <number>[optional]\n", argv[0]);
      exit(EXIT_FAILURE);
  }
    if (pipe(pipefd1) == -1) {
        perror("pipe");
        exit(EXIT_FAILURE);
    }
    child = fork();  /* fork now */
  if (child == 0) {
  /* Child process: IO redirection*/
    CLOSE(pipefd1[0]);
    IOredir(pipefd1[1],STDOUT_FILENO);/*write to pipefd1*/
    if(argc==1){
      char* argv1[2];
      argv1[0]="/Users/scott/Documents/CDT/Pipeline/src/src/wordgen";
      argv1[1]=NULL;
      if (execvp (argv1[0],argv1)==-1) {
        fprintf (stderr, "ERROR-->execv failure for[wordgen]: %s\n", strerror (errno));
        exit(EXIT_FAILURE);
      }
    }else{
      char* argv1[3];
      argv1[0]="/Users/scott/Documents/CDT/Pipeline/src/src/wordgen";
      argv1[1]=argv[1];
      argv1[2]=NULL;
      if (execvp (argv1[0],argv1)==-1) {
```

```
70          fprintf (stderr, "ERROR-->execv␣failure␣for[wordgen]:␣%s\n", strerror (errno));
71          exit(EXIT_FAILURE);
72        }
73      }
74      exit(EXIT_FAILURE);/*we should never reach here*/
75    } else if (child < 0) {
76      fprintf (stderr, "ERROR-->fork␣failure␣for␣[%s]:␣%s\n", "wordgen", strerror (errno));
77      return EXIT_FAILURE;
78    } else {/* Parent process*/
79      if (pipe(pipefd2) == -1) {
80        perror("Pipe");
81        exit(EXIT_FAILURE);
82      }
83      grandchild=fork(); /*launch the wordsearch*/
84      if (grandchild == 0) {
85        CLOSE(pipefd1[1]);
86        CLOSE(pipefd2[0]);
87        IOredir(pipefd1[0],STDIN_FILENO);  /*connect the Pipe*/
88        IOredir(pipefd2[1],STDOUT_FILENO);
89        char *a[3];
90        a[0]="./wordsearch";
91        a[1]="wordlist.txt";
92        a[2]=NULL;
93        if (execvp ("/Users/scott/Documents/CDT/Pipeline/src/src/wordsearch",a)==-1) { /*exec
              */
94            fprintf (stderr, "ERROR-->execv␣failure␣for[wordsearch]:␣%s\n", strerror (errno)
                  );
95            exit(EXIT_FAILURE);
96        }
97      } else if(grandchild<0){
98        fprintf (stderr, "ERROR-->fork␣failure␣for␣[wordgen]:␣%s\n", strerror (errno));
99        return EXIT_FAILURE;
100     }else{
101     /* Parent process*/
102       greatgrandchild=fork();
103       if (greatgrandchild == 0) {
104         CLOSE(pipefd1[0]);
105         CLOSE(pipefd1[1]);
106         CLOSE(pipefd2[1]);
107         IOredir(pipefd2[0],STDIN_FILENO);
108         char* argv2[2];
109         argv2[0]="/Users/scott/Documents/CDT/Pipeline/src/src/pager";
110         argv2[1]=NULL;
111         if (execvp ("/Users/scott/Documents/CDT/Pipeline/src/src/pager",argv2)==-1) { /*exec
              */
112           fprintf (stderr, "ERROR-->execv␣failure␣for[pager]:␣%s\n", strerror (errno));
113           exit(EXIT_FAILURE);
114         }
115       }else if(greatgrandchild<0){
116           fprintf (stderr, "ERROR-->fork␣failure␣for␣[pager]:␣%s\n", strerror (errno));
117           exit(EXIT_FAILURE);
118       }else{
119         /*close all the file discripters in Mother*/
120         CLOSE(pipefd1[0]);
121         CLOSE(pipefd1[1]);
122         CLOSE(pipefd2[0]);
123         CLOSE(pipefd2[1]);
124         pid_t children;
125         while((children = waitpid(-1,&waitStatus,0))>0 && children!=1){
126           if (WIFEXITED(waitStatus)) {
127             fprintf(stderr,"pid␣%d␣exited,␣status=%d\n",children,WEXITSTATUS(waitStatus));
128           }else if (WIFSIGNALED(waitStatus)) {
129             if(WCOREDUMP(waitStatus))
130               fprintf(stderr,"pid␣%d␣exited,␣killed␣by␣signal␣with␣coredump[run␣\"ulimit␣-c␣
                      unlimited\"]␣%d\n",children,WTERMSIG(waitStatus));
131             else
132               fprintf(stderr,"pid␣%d␣exited␣killed␣by␣signal␣%d\n", children, WTERMSIG(
                      waitStatus));
```

```
133             }else if (WIFSTOPPED(waitStatus)) {
134               fprintf(stderr,"pid␣%d␣exited␣stopped␣by␣signal␣%d\n",children, WSTOPSIG(
                      waitStatus));
135             }
136           }
137           if(errno == EINTR && children<0){
138             fprintf(stderr,"\nwaitpid␣failure␣by␣signal␣interuption␣by␣calling␣process:%s\n",
                      strerror(errno));
139             exit(EXIT_FAILURE);
140           }
141         }
142       }
143     }
144     return EXIT_SUCCESS;
145   }
```

2 Experimental Screenshots

```
blablall:src scott$ ./wordgen 1000 | wc
     1000    1000    7468
blablall:src scott$ time ./wordgen 1000000  >/dev/null

real    0m1.189s
user    0m0.714s
sys     0m0.004s
blablall:src scott$ echo "COMPUTER"| ./wordsearch wordlist.txt
67529 words Accepted, 2376 words Rejected
COMPUTER
Mateched:1 Words
blablall:src scott$ time ./wordgen 10000 | ./wordsearch  wordlist.txt >/dev/null
67529 words Accepted, 2376 words Rejected
Mateched:726 Words

real    0m0.079s
user    0m0.071s
sys     0m0.012s
blablall:src scott$
```

Figure 1: wordgen.c&wordsearch.c

```
blablall:src scott$ ./pager <wordlist.txt
a
a-horizon
a-ok
aardvark
aardwolf
ab
aba
abaca
abacist
aback
abactinal
abacus
abaddon
abaft
abalienate
abalienation
abalone
abampere
abandon
abandoned
abandonment
abarticulation
abase
abased
                ---Press RETURN for more---
abasement
abash
abashed
abashment
abasia
abasic
abate
abatement
abating
abatis
abatjour
abattis
abattoir
abaxial
abba
abbacy
abbatial
abbatical
abbatis
abbe
abbess
abbey
abbot
                ---Press RETURN for more--- q
*** Pager terminated by Q command ***
blablall:src scott$ echo $?
0
```

Figure 2: pager.c

```
[blablall:src scott$ ./launcher 100                                    ]
pid 13174 exited, status=0
67529 words Accepted, 2376 words Rejected
Mateched:13 Words
ELDSFUOLP
pid 13175 exited, status=0
MGLUDEA
BABPHBATIB
ASSTK
OMPWMKE
VXQO
IUCNYRB
WQAUYKI
QENXXO
MASUGGTETK
SINBQO
DZJ
SONPRWWXRM
pid 13176 exited, status=0
[blablall:src scott$ ./launcher                                        ]
67529 words Accepted, 2376 words Rejected
YFWO
UJVZQ
QXKTVVT
PBKFSTOSQP
RFRUMVY
NWC
GOTJVGE
CLOCVFPU
GQYL
QUBMOLNFC
WSYLVMMLL
TTJEL
EGUINHYB
IJFAO
LWYT
TGXZC
QJRLIZT
YEAIID
DZLLNWCYSN
XXICFR
WSKHBWAVHL
MSEHAH
CSG
PVCI
                ---Press RETURN for more---
```

Figure 3: launcher.c 100

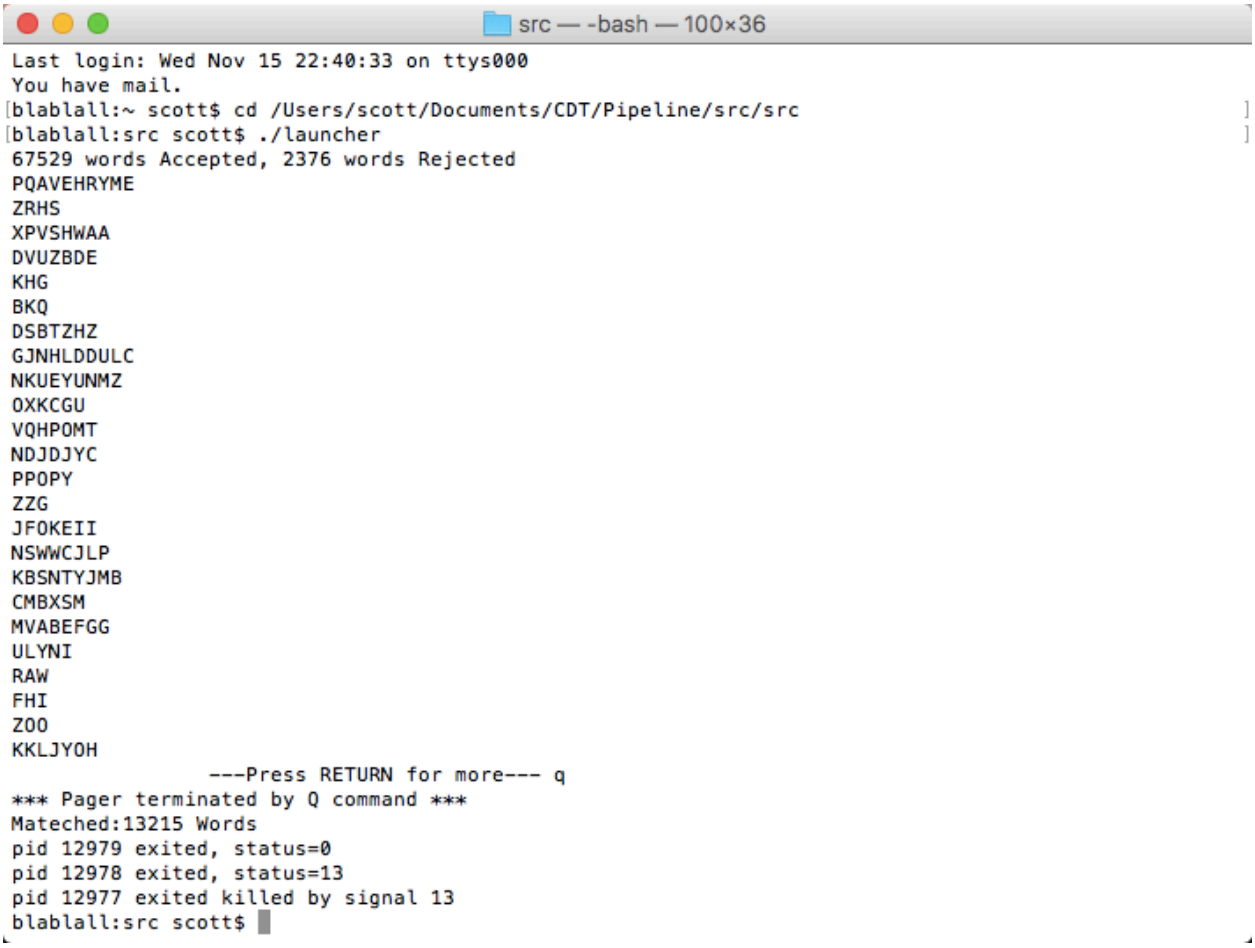

Figure 4: launcher.c