Donghyun Park

Prof. Hakner

ECE 357

10/29/2017

PSet 4: Piping

wordgen.c

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

#include <limits.h>

#include <string.h>

int main(int argc, char \*\*argv){

int wordCount, length, totalNum;

char nc[11]; //10 char + null terminator

time\_t t;

srand((unsigned) time(&t));

if(argc > 1){

if(!strcmp(argv[1], "0")){

wordCount = INT\_MAX; //specifying 0 results in 'infinite' word generation

}else{

wordCount = atoi(argv[1]);

}

}else{

wordCount = INT\_MAX; //no argument results in 'infinite' word generation

}

totalNum = wordCount;

for(; wordCount > 0; wordCount--){

length = 3 + (rand() % 8); //min 3 characters in a word, 0-7 possible numbers, last index reserved for null terminator

for(int i = 0; i < length; i++){

nc[i] = (char)(65 + (rand() % 26));

}

nc[length] = '\0';

printf("%s\n", nc);

}

fprintf(stderr, "Finished generating %d candidate words\n", totalNum);

return 0;

}

wordsearch.c

#include <stdlib.h>

#include <stdio.h>

#include <errno.h>

#include <string.h>

#include <ctype.h>

#include <signal.h>

int matches;

void sigHandle(){

fprintf(stderr, "Matched %d words\n", matches);

exit(0);

}

void upperCase(char \*word){

for(int h = 0; h < strlen(word); h++){

word[h] = toupper(word[h]);

}

}

int main(int argc, char \*\*argv){

FILE \*inputFile;

size\_t len1 = 0, len2 = 0;

ssize\_t read;

char \*buf[250000], \*line = NULL;

int i = 0;

signal(SIGPIPE, sigHandle);

if(argc < 2){

fprintf(stderr, "Error: No intput file specified\n");

return -1;

}else{

if((inputFile = fopen(argv[1], "r")) == NULL){

fprintf(stderr, "Error: Failed to open file [%s] - %s\n", argv[1], strerror(errno));

return -1;

}

}

while((read = getline(&buf[i], &len1, inputFile)) != -1){

upperCase(buf[i]);

i++;

}

while((read = getline(&line, &len2, stdin)) != -1){

upperCase(line);

for(int j = 0; j < i; j++){

if(!strcmp(line, buf[j])){

printf("%s", line);

matches++;

break;

}

}

}

for(int k = 0; k < i; k++){

free(buf[k]);

}

printf("Matched %d words\n", matches);

fclose(inputFile);

return 0;

}

pager.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <errno.h>

int main(int argc, int \*\*argv){

ssize\_t read;

size\_t len = 0;

char \*command, \*line = NULL;

int count = 0;

int c;

FILE \*dev;

while((read = getline(&line, &len, stdin)) != -1){

printf("%s", line);

count++;

if(count == 23){

printf("---Press RETURN for more---");

if((dev = fopen("/dev/tty", "r+")) == NULL){

fprintf(stderr, "Error: Failed to open file [/dev/tty] - %s\n", strerror(errno));

return -1;

}

c = getc(dev);

if(c == 113 || c == 81){

exit(0);

}

count = 0;

fclose(dev);

}

}

return 0;

}

launcher.c

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include <unistd.h>

#include <fcntl.h>

#include <sys/wait.h>

#include <sys/types.h>

#include <sys/time.h>

#include <sys/resource.h>

#include <errno.h>

int main(int argc, char \*\*argv){

int pipe1[2], pipe2[2], status; //0 read, 1 write

pid\_t cpid1, cpid2, cpid3, wpid1, wpid2, wpid3;

struct rusage ru;

if(pipe(pipe1) == -1){ //wordgen->wordsearch

fprintf(stderr, "Failed to create pipe1 - %s\n", strerror(errno));

return -1;

}

if(pipe(pipe2) == -1){ //wordsearch->pager

fprintf(stderr, "Failed to create pipe2 - %s\n", strerror(errno));

return -1;

}

switch(cpid1 = fork()){ //exec wordgen

case -1:

fprintf(stderr, "Error: Failed to fork() process 1 'wordgen' - %s\n", strerror(errno));

exit(1);

break;

case 0: //child process, dup stdout(1) of wordgen to write/input of pipe1(pipe1[1])

if(dup2(pipe1[1], 1) == -1){

fprintf(stderr, "Error: Failed to dup2() fd 1 to input of pipe1 - %s\n", strerror(errno));

return -1;

}

if(close(pipe1[0]) == -1){

fprintf(stderr, "Error: Failed to close read end of pipe1 - %s\n", strerror(errno));

}

if(close(pipe1[1]) == -1){

fprintf(stderr, "Error: Failed to close write end of pipe1 - %s\n", strerror(errno));

}

if(close(pipe2[0]) == -1){

fprintf(stderr, "Error: Failed to close read end of pipe2 - %s\n", strerror(errno));

}

if(close(pipe2[1]) == -1){

fprintf(stderr, "Error: Failed to close write end of pipe2 - %s\n", strerror(errno));

}

if(execlp("./wordgen", "wordgen", argv[1], (char \*)NULL) == -1){

fprintf(stderr, "Error: Failed to execute ./wordgen - %s\n", strerror(errno));

}

break;

default: //parent process

switch(cpid2 = fork()){ //exec wordsearch

case -1:

fprintf(stderr, "Error: Failed to fork() process 2 'wordsearch' - %s\n", strerror(errno));

exit(1);

break;

case 0: //child process, dup read/output of pipe1(pipe1[0]) to stdin(0) of wordsearch, stdout(1) of wordsearch to write/intput of pipe2(pipe2[1])

if(dup2(pipe1[0], 0) == -1){

fprintf(stderr, "Error: Failed to dup2() fd 0 to output of pipe1 - %s\n", strerror(errno));

return -1;

}

if(dup2(pipe2[1], 1) == -1){

fprintf(stderr, "Error: Failed to dup2() fd 1 to input of pipe2 - %s\n", strerror(errno));

return -1;

}

if(close(pipe1[0]) == -1){

fprintf(stderr, "Error: Failed to close read end of pipe1 - %s\n", strerror(errno));

}

if(close(pipe1[1]) == -1){

fprintf(stderr, "Error: Failed to close write end of pipe1 - %s\n", strerror(errno));

}

if(close(pipe2[0]) == -1){

fprintf(stderr, "Error: Failed to close read end of pipe2 - %s\n", strerror(errno));

}

if(close(pipe2[1]) == -1){

fprintf(stderr, "Error: Failed to close write end of pipe2 - %s\n", strerror(errno));

}

if(execlp("./wordsearch", "wordsearch", "dictionary.txt", (char \*)NULL) == -1){

fprintf(stderr, "Error: Failed to execute ./wordsearch - %s\n", strerror(errno));

}

break;

default: //parent process

switch(cpid3 = fork()){ //exec pager

case -1:

fprintf(stderr, "Error: Failed to fork() process 3 'pager' - %s\n", strerror(errno));

exit(1);

break;

case 0: //child process, dup read/output of pipe2(pipe[0]) to stdin(0) of pager

if(dup2(pipe2[0], 0) == -1){

fprintf(stderr, "Error: Failed to dup2() fd 0 to output of pipe2 - %s\n", strerror(errno));

return -1;

}

if(close(pipe1[0]) == -1){

fprintf(stderr, "Error: Failed to close read end of pipe1 - %s\n", strerror(errno));

}

if(close(pipe1[1]) == -1){

fprintf(stderr, "Error: Failed to close write end of pipe1 - %s\n", strerror(errno));

}

if(close(pipe2[0]) == -1){

fprintf(stderr, "Error: Failed to close read end of pipe2 - %s\n", strerror(errno));

}

if(close(pipe2[1]) == -1){

fprintf(stderr, "Error: Failed to close write end of pipe2 - %s\n", strerror(errno));

}

if(execlp("./pager", "./pager", (char \*)NULL) == -1){

fprintf(stderr, "Error: Failed to execute ./pager - %s\n", strerror(errno));

}

break;

}

break;

}

break;

}

if(close(pipe1[0]) == -1){

fprintf(stderr, "Error: Failed to close read end of pipe1 - %s\n", strerror(errno));

}

if(close(pipe1[1]) == -1){

fprintf(stderr, "Error: Failed to close write end of pipe1 - %s\n", strerror(errno));

}

if(close(pipe2[0]) == -1){

fprintf(stderr, "Error: Failed to close read end of pipe2 - %s\n", strerror(errno));

}

if(close(pipe2[1]) == -1){

fprintf(stderr, "Error: Failed to close write end of pipe2 - %s\n", strerror(errno));

}

if((wpid1 = wait3(&status, 0, &ru)) < 0){

fprintf(stderr, "Error: Failed to wait() process on pid=[%d] - %s\n", cpid1, strerror(errno));

}else{

fprintf(stderr, "Child 1 %d exited with %d\n", wpid1, status);

}

if((wpid2 = wait3(&status, 0, &ru)) < 0){

fprintf(stderr, "Error: Failed to wait() process on pid=[%d] - %s\n", cpid2, strerror(errno));

}else{

fprintf(stderr, "Child 2 %d exited with %d\n", wpid2, status);

}

if((wpid3 = wait3(&status, 0, &ru)) < 0){

fprintf(stderr, "Error: Failed to wait() process on pid=[%d] - %s\n", cpid3, strerror(errno));

}else{

fprintf(stderr, "Child 3 %d exited with %d\n", wpid3, status);

}

return 0;

}

Shell Interpreter Scripts:

testme.sh:

#!/home/dhpark/school/OS/OSHW3/myshell

#This is an example of a shell script that your shell must execute correctly

#notice that lines starting with a # sign are ignored as comments!

#let’s say this here file is called testme.sh. you created it with say

#vi testme.sh ; chmod +x testme.sh

#you invoked it with

#./testme.sh

cat >cat.out

#at this point, type some lines at the keyboard, then create an EOF (Ctrl-D)

#your shell invoked the system cat command with output redirected to cat.out

cat cat.out

#you better see the lines that you just typed!

exit 123

#after your shell script exits, type echo $? from the UNIX system shell

#the value should be 123. Since your shell just exited, the following

#bogus command should never be seen

test2.sh:

#!/home/dhpark/school/OS/OSHW3/myshell

#here is another example, say it is called test2.sh

#you invoked it with

#./test2.sh <input.txt

cat >cat2.out

#since you invoked the shell script (via the system shell such as bash)

#with stdin redirected, your shell runs cat which gets stdin from input.txt

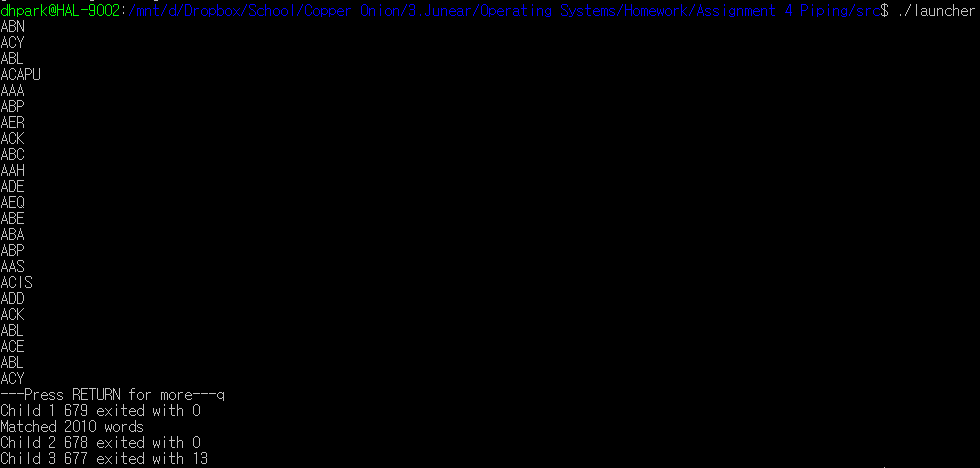
exit

#the above exit had no specified return value, so your shell exited with 0

#again, test this with echo $?

Sample Run

Infinite wordgen with signal handling:



Finite wordgen:

