**Problem Set 4 – A Three-Command Pipeline**

1. **Source Code - launcher.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#include <sys/wait.h>

void closePipes(int \*pipe1, int \*pipe2);

void waitfork(pid\_t cpid, int stat);

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

pid\_t gen, search, page;

int fd1[2], fd2[2];

int status;

char \*arggen[] = {"./wordgen", argv[1], NULL};

char \*argsearch[] = {"./wordsearch", "wordlist\_small.txt", NULL};

char \*argpage[] = {"./pager", NULL};

if(pipe(fd1) == -1 || pipe(fd2) == -1){

fprintf(stderr, "Error creating pipe: %s\n", strerror(errno));

exit(EXIT\_FAILURE);

}

switch(gen = fork()){

case -1:

fprintf(stderr, "Error creating fork for wordgen: %s\n", strerror(errno));

exit(EXIT\_FAILURE);

case 0:

if(dup2(fd1[1], STDOUT\_FILENO) < 0){

fprintf(stderr, "Error dup2 in first pipe connection: %s\n", strerror(errno));

exit(EXIT\_FAILURE);

}

closePipes(fd1, fd2);

execvp(arggen[0], arggen);

fprintf(stderr, "Error occured executing wordgen: %s\n", strerror(errno));

break;

default:

switch(search = fork()){

case -1:

fprintf(stderr, "Error creating fork for wordsearch: %s\n", strerror(errno));

exit(EXIT\_FAILURE);

case 0:

if(dup2(fd1[0], STDIN\_FILENO) < 0 || dup2(fd2[1], STDOUT\_FILENO) <0){

fprintf(stderr, "Error dup2 in second pipe connection: %s\n", strerror(errno));

exit(EXIT\_FAILURE);

}

closePipes(fd1,fd2);

execvp(argsearch[0], argsearch);

fprintf(stderr, "Error occured executing wordsearch: %s\n", strerror(errno));

break;

default:

switch(page = fork()){

case -1:

fprintf(stderr, "Error creating fork for pager: %s\n", strerror(errno));

exit(EXIT\_FAILURE);

case 0:

if(dup2(fd2[0], STDIN\_FILENO) < 0){

fprintf(stderr, "Error dup2 in pager: %s\n", strerror(errno));

exit(EXIT\_FAILURE);

}

closePipes(fd1,fd2);

execvp(argpage[0], argpage);

fprintf(stderr, "Error occured executing pager: %s\n", strerror(errno));

break;

}

}

}

closePipes(fd1, fd2);

waitfork(page, status);

waitfork(search, status);

waitfork(gen, status);

exit(EXIT\_SUCCESS);

}

void closePipes(int \*pipe1, int \*pipe2){

if(close(pipe1[0]) < 0 || close(pipe1[1]) < 0)

{

fprintf(stderr, "Error closing pipe 1: %s", strerror(errno));

exit(EXIT\_FAILURE);

}

if(close(pipe2[0]) < 0 || close(pipe2[1]) < 0)

{

fprintf(stderr, "Error closing pipe 2: %s", strerror(errno));

exit(EXIT\_FAILURE);

}

}

void waitfork(pid\_t cpid, int stat){

do{

if(waitpid(cpid, &stat, 0) == -1){

fprintf(stderr, "Error waiting cpid %i: %s\n",cpid, strerror(errno));

exit(EXIT\_FAILURE);

}

if(WIFEXITED(stat)){

printf("Child %i exited with %i\n", cpid, stat);

}else if(WIFSIGNALED(stat)){

printf("Child %i killed by signal %d\n", cpid, WTERMSIG(stat));

}

}while(!WIFEXITED(stat) && !WIFSIGNALED(stat));

}

1. **Source Code - wordgen.c**

#include <stdio.h>

#include <stdlib.h>

#include <errno.h>

#include <string.h>

#include <time.h>

static char \*rand\_string(char \*str, size\_t size);

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

char \*line;

int num = 0;

size\_t size = 5;

srand(time(NULL));

if(argv[1] != NULL) num = atoi(argv[1]);

if(!num){

fprintf(stderr, "No input argument. Generating words in an endless loop.\n");

while(1){

line = rand\_string(line, size - 2);

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

}

}else{

int i = 0;

while(i++ < num){

line = rand\_string(line, size - 2);

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

}

fprintf(stderr, "Finished generating %i candidate words\n", num);

}

return 0;

}

static char \*rand\_string(char \*str, size\_t size){

int n = 0, max = 0;

time\_t t;

max = max + rand() % ((int) size) + 3;

while (n < max) {

char c = (rand () % 26 + 65);

str[n++] = c;

}

str[n++] = '\0';

return str;

}

1. **Source Code - wordsearch.c**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

#include <sys/wait.h>

int upper\_string(char \*s);

void sigpipeHandler(int sig);

int accepted = 0;

int rejected = 0;

int matched = 0;

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

char \*dic[500000];

char buf[BUFSIZ];

int index = 0;

size\_t len = 0;

ssize\_t rd;

pid\_t cpid;

int fd[2];

if (argc < 2){

fprintf(stderr, "Not enough input arguments. Specify dictionary.\n");

exit(EXIT\_FAILURE);

}

signal(SIGPIPE, sigpipeHandler);

FILE \*f;

f = fopen(argv[1], "r");

if(!f){

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

return -1;

}

while(rd = getline(&dic[index], &len, f) != -1) {

if(upper\_string(dic[index])) {

accepted++;

index++;

}else{

rejected++;

}

}

if(rd == -1){

fprintf(stderr, "Error reading from file %s: %s\n", argv[1], strerror(errno));

exit(EXIT\_FAILURE);

}

fprintf(stderr, "Accepted %i words, rejected %i words\n", accepted, rejected);

while ((fgets(buf, sizeof buf, stdin) != NULL) && (buf[0] != '\n')){

index = 0;

while(index < accepted){

if(strcmp(buf, dic[index++]) == 0) {

printf("%s", buf);

matched ++;

}

}

}

fprintf(stderr, "Matched %i words\n", matched);

index = 0;

while(index<accepted){

free(dic[index++]);

}

fclose(f);

exit(EXIT\_SUCCESS);

return 0;

}

int upper\_string(char \*s) {

int c = 0;

while (s[c] != '\0') {

if (s[c] >= 'a' && s[c] <= 'z') {

s[c] = s[c] - 32;

}else{

if(s[c] < 'A' || s[c] > 'Z') {

if(s[c] != '\n') return 0;

}

}

c++;

}

s[c] = '\0';

return 1;

}

void sigpipeHandler(int sig){

fprintf(stderr, "Program interrupted by SIGPIPE, at the point of termination, Matched %i words\n", matched);

exit(EXIT\_FAILURE);

}

1. **Source Code - pager.c**

#include <stdio.h>

#include <fcntl.h>

#include <stdlib.h>

#include <unistd.h>

#include <errno.h>

#include <string.h>

#include <sys/types.h>

int main(){

int numline = 0;

char buf[BUFSIZ];

char cmd[BUFSIZ];

size\_t len = 0;

ssize\_t fd;

while (fgets(buf, sizeof buf, stdin) != NULL)

{

numline++;

printf("%s", buf);

if(numline == 23){

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

cmd[0] = '\0'; //flushes cmd

fd = open("/dev/tty", O\_RDONLY);

read(fd, cmd, sizeof cmd);

close(fd);

if(cmd[0] == 'Q' || cmd[0] == 'q'){

printf("\*\*\* Pager terminated by Q command \*\*\*\n");

exit(EXIT\_SUCCESS);

}

if(cmd[0] == '\0'){

printf("\*\*\* Pager terminated by EOF \*\*\*\n");

exit(EXIT\_SUCCESS);

}

numline = 0;

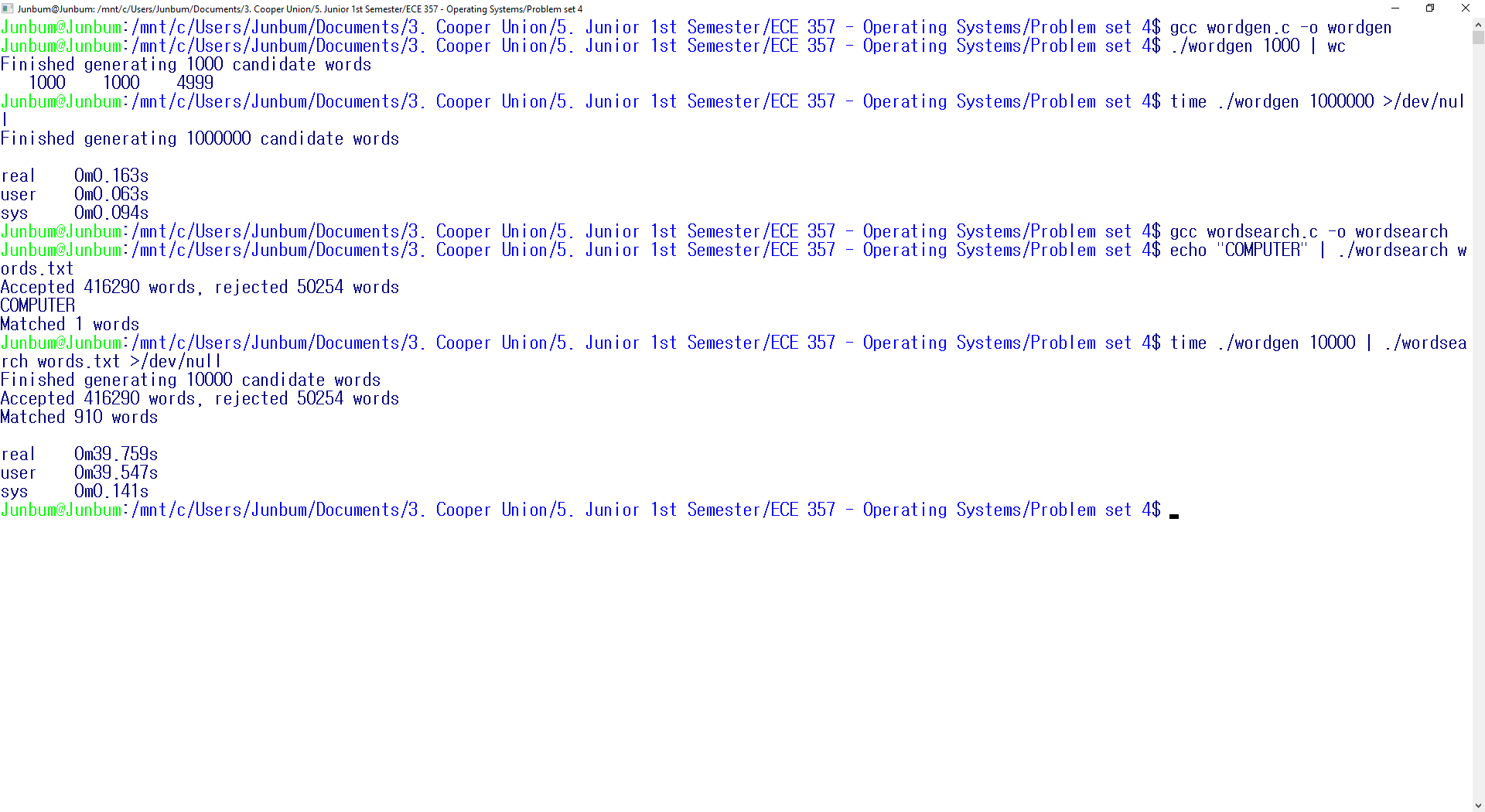
}

}

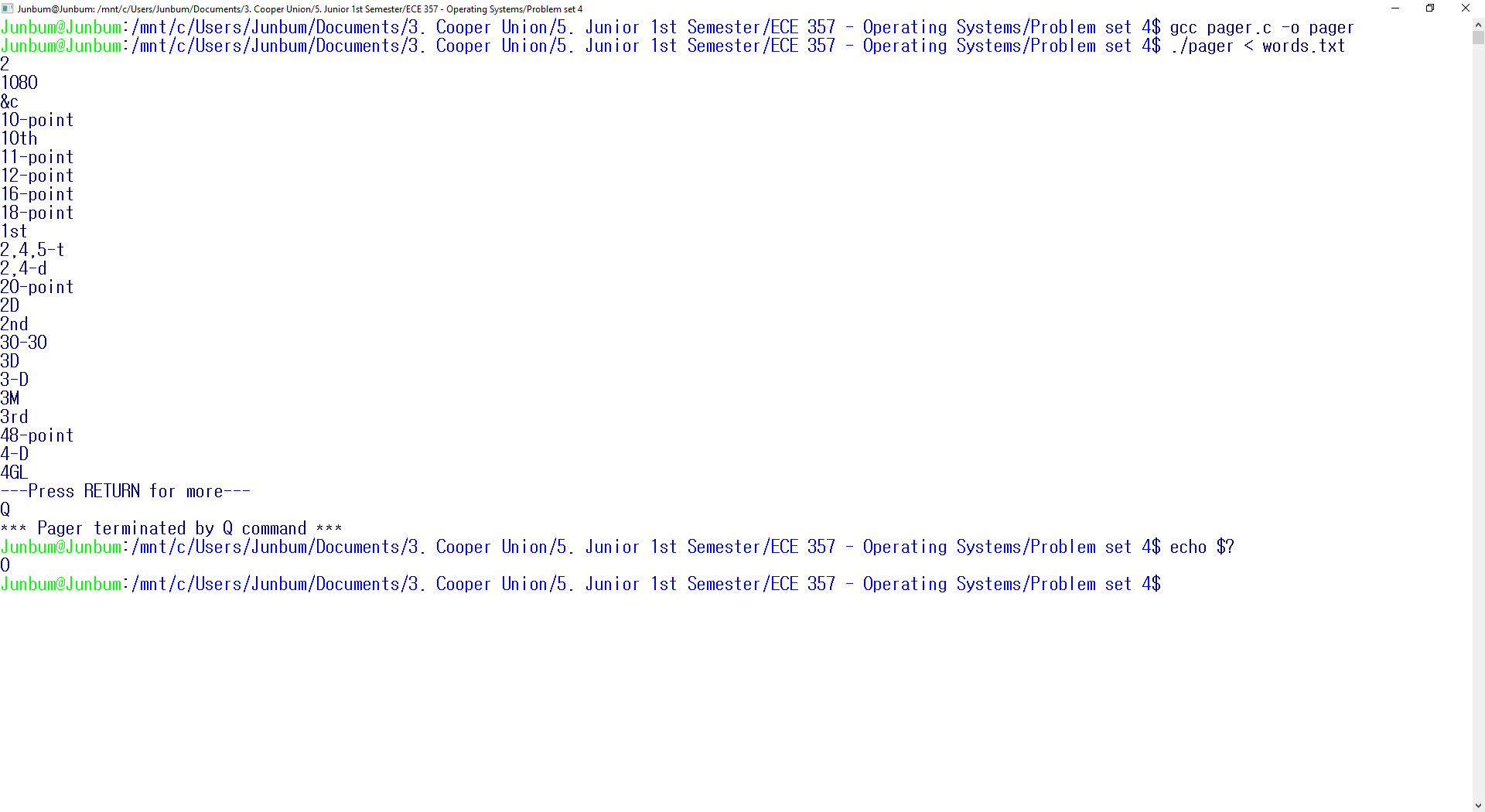
return 0;

}

1. **Screenshot of a Sample Run**
2. ­Sample run for wordgen and wordsearch



1. Sample run for pager



1. Sample run for launcher

