­­­­CS 4540 – Operating Systems

Instructor: Lilien, Western Michigan University

Justin Lanyon

**SOFTWARE LIFE CYCLE REPORT – FOR LAB ASSIGNMENT 02**

**PHASE 1: SPECIFICATION (“What do we build?”)**

Design a program using ordinary pipes in which there are two processes (a parent and a child). The parent process asks the user to enter a candidate password for his “username” in order to access Application X. (We do not care what application is X.) In response, the parent sends the password to the child. The child checks the strength of the user's password, and returns one of the following three decisions:

STRONG PASSWORD If the password has more than 8 characters and contains uppercase and lower-case letters as well as numbers. Examples: OPSlmr183, 187jklPNR, etc.

WEAK PASSWORD If the password has more than 8 characters and does not mix upper-case letters, lower-case letters, and numbers. Examples: AAAAAAAAA, 123456789, aaaaaacccc, EERReeeee, 7878EEEE, hhhh4545, etc.

SHORT PASSWORD If the password has fewer than 8 characters.

The program requires using two pipes: one for sending the password from the parent to the child, and the other for sending the child’s decision to the parent.

**PHASE 2: DESIGN**

**2.1 Modules and Their Basic Structure**

Main function – Handles the forking and process management

getPass function – gets the password from the user within the parent process

passTest function – determines the passwords strength with the given criteria within the child process

**Phase 3: RISK ANALYSIS**

There are no risks (to cost, human health, timetable) are identifiable.

**PHASE 4: VERIFICATION**

This algorithm has a very standard path that has been verified for its correctness by me through analyzing its steps in comparison to the specifications.

**PHASE 5: CODING**

**5a) Code Refinement #1(class structure with pseudocode only;)**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Course: CS 4540 – Fall 2015**

**\* Assignment 2 - Problem 1**

**\* Name: Justin Lanyon**

**\* E-mail: justin.j.lanyon@wmich.edu**

**\* Submitted: 10/19/2015**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**//Initialize variables**

**//create parent & child process**

**//parent-get password**

**//child – test password strength return strength**

**5b) Code Refinement #2 (complete program--with complete fields/properties, code for constructor/methods)**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Course: CS 4540 – Fall 2015**

**\* Assignment 2 - Problem 1**

**\* Name: Justin Lanyon**

**\* E-mail: justin.j.lanyon@wmich.edu**

**\* Submitted: 10/19/2015**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**//Main Method**

**//fork process**

**//child process**

**//wait until written to**

**//process password**

**//return strength**

**//parent process**

**//receive password**

**//pipe to child process**

**//wait until strength is piped back**

**//output password strength**

**//end**

**5c) Code Refinement #3**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Course: CS 4540 – Fall 2015**

**\* Assignment 2 - Problem 1**

**\* Name: Justin Lanyon**

**\* E-mail: justin.j.lanyon@wmich.edu**

**\* Submitted: 10/19/2015**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**//Main Method**

**//fork process**

**//child process**

**//wait until written to**

**//process password using passCheck**

**//return strength**

**//parent process**

**//receive password from getPass function**

**//pipe to child process**

**//wait until strength is piped back**

**//output password strength**

**//end**

**//passCheck Function – test password strength based on criteria**

**//getPass function – receive password from stdin**

**5d) Code Refinement #4 (complete program--with complete fields/properties, code for constructor/methods)**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* Course: CS 4540 – Fall 2015**

**\* Assignment 2 - Problem 1**

**\* Name: Justin Lanyon**

**\* E-mail: justin.j.lanyon@wmich.edu**

**\* Submitted: 10/19/2015**

**/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**// Parent: reads from P1\_READ, writes on P1\_WRITE**

**// Child: reads from P2\_READ, writes on P2\_WRITE**

**#define P1\_READ 0**

**#define P2\_WRITE 1**

**#define P2\_READ 2**

**#define P1\_WRITE 3**

**#define NUM\_PIPES 2**

**#define false 0 //Used for testing password strength**

**//Prototypes for other functions**

**char \* passTest(char \* password);**

**char \* getPass(char \* password);**

**int main(int argc, char \*argv[])**

**{**

**int fd[2\*NUM\_PIPES];**

**int len, i;**

**char \* val = malloc(100);**

**pid\_t pid;**

**int tend =100;**

**// create all the descriptor pairs we need**

**for (i=0; i<NUM\_PIPES; ++i)**

**{**

**if (pipe(fd+(i\*2)) < 0)**

**{**

**perror("Piping Failed");**

**exit(EXIT\_FAILURE);**

**}**

**}**

**//FORK ERROR**

**if ((pid = fork()) < 0)**

**{**

**perror("Fork Error");**

**return EXIT\_FAILURE;**

**}**

**//child process**

**if (pid == 0)**

**{**

**// Child. CLOSE UNNEEDED FD**

**close(fd[P1\_READ]);**

**close(fd[P1\_WRITE]);**

**// used for output**

**pid = getpid();**

**// wait until parent sends a value**

**len = read(fd[P2\_READ], val, tend);**

**if (len < 0)**

**{**

**perror("Child: Failed to read data from pipe\n");**

**exit(EXIT\_FAILURE);**

**}**

**else if (len == 0)//In case nothing in pipe**

**{**

**fprintf(stderr, "Child: Read EOF from pipe\n");**

**}**

**else**

**{**

**// report what was recieved**

**//printf("Child(%d): Received >%s<\n", pid, val);**

**val = passTest(val); //CHECK PASSWORD**

**tend = strlen(val);**

**//printf("Child(%d): Sending >%s< back\n", pid, val);**

**if (write(fd[P2\_WRITE], val, tend) < 0)**

**{**

**perror("Child: Failed to write response value");**

**exit(EXIT\_FAILURE);**

**}**

**}**

**// finished. close remaining descriptors.**

**close(fd[P2\_READ]);**

**close(fd[P2\_WRITE]);**

**return EXIT\_SUCCESS;**

**//END CHILD PROCESS**

**}else{**

**//PARENT PROCESS**

**// Close unused FD**

**close(fd[P2\_READ]);**

**close(fd[P2\_WRITE]);**

**// used for output**

**pid = getpid();**

**// send password to child**

**val = getPass(val);**

**strtok(val, "\n");**

**tend = strlen(val);**

**//printf("Parent(%d): Sending %s to child\n", pid, val);**

**if (write(fd[P1\_WRITE], val, tend) != tend)**

**{**

**perror("Parent: Failed to send value to child ");**

**exit(EXIT\_FAILURE);**

**}**

**// now wait for a response**

**len = read(fd[P1\_READ], val, tend);**

**if (len < 0)**

**{**

**perror("Parent: failed to read value from pipe\n");**

**exit(EXIT\_FAILURE);**

**}**

**else if (len == 0)**

**{**

**// not an error, but certainly unexpected**

**fprintf(stderr, "Parent(%d): Read EOF from pipe\n", pid);**

**}**

**else**

**{**

**// report what we received**

**//printf("Parent(%d): Received %s\n", pid, val);**

**printf("That is a %s password\n",val );**

**}**

**// close descriptors**

**close(fd[P1\_READ]);**

**close(fd[P1\_WRITE]);**

**// wait for child termination**

**wait(NULL);**

**return EXIT\_SUCCESS;**

**}**

**}//END MAIN**

**char \* getPass(char \* password){**

**printf("\nWELCOME TO APPLICATION X\n");**

**printf("Please enter the password: ");**

**fgets(password, 50, stdin);**

**return password;**

**}**

**char \* passTest(char \* password){**

**char \* passStrength = "";**

**int i=0;**

**int found\_lower = false, found\_upper = false, found\_number =false;**

**while(password[i] != '\0' || !(found\_lower && found\_upper&& found\_number)) {**

**found\_lower = found\_lower || (password[i] >= 'a' && password[i] <= 'z');**

**found\_upper = found\_upper || (password[i] >= 'A' && password[i] <= 'Z');**

**found\_number = found\_number ||(password[i] >= '0' && password[i] <= '9');**

**i++;**

**}**

**if(found\_lower && found\_upper&& found\_number){**

**strcpy(passStrength, "strong");**

**}else{**

**strcpy(passStrength, "weak");**

**}**

**if(strlen(password)<8){**

**strcpy(passStrength, "short");**

**}**

**printf("passStrength >%s<\n",passStrength);**

**return passStrength;**

**}**

**PHASE 6: TESTING**

One can test it simply by selecting different input files for each run and ensuring it produces the proper output.

**PHASE 7: REFINING THE PROGRAM**

No refinements are needed. In this program, I have already included all required features!

**PHASE 8: PRODUCTION**

I prepared a copy of the entire program for Lab TA’s evaluation, as specified by the TA.Then, I sent

electronically the copy to the Lab TA.

**PHASE 9: MAINTENANCE**

To fully benefit from the program evaluation feedback received from the Lab TA, I will perform program

maintenance. This means that I should use all TAs feedback to improve my program.