**ASSIGNMENT - 2**

**SPECIAL PROBLEMS IN COMPUTER SCIENCE: SECURITY MINING BIGDATA WITH HIGH –PERFORMANCE COMPUTING**

**COURSE CODE-CS-5331**

**TEAM MEMBERS:**

**1.NAVEEN AJAY KUMAR NAGINENI –R11840700**

**2.BHUVAN RISHITHA ARAVAL PAPOLU-R11850996**

**3.VENKATA KRISHNA SAI PAMIDIMARRI-R11848326**

**Code:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**/\***

**#include <sys/wait.h> \*/**

**#include <fcntl.h>**

**#include <readline/readline.h>**

**#include <readline/history.h>**

**#include "parse.h"**

**/\* In above we take all the required libraries to execute the desired developement code\*/**

**/\* Initiating arguments Run and arguments are NNN, Command \*/**

**void Initiate(Command);**

**void Run(NNN \*, Command, int, int);**

**void Belt(char \*);**

**int done = 0; /\* when variable done is equal to zero, it is initiating Globally and This globally results when user done with the program\*/**

**/\* Stating Find function and placing Mark as variable as an argument \*/**

**static void Find(int Mark) {**

**putchar('\n');**

**return;**

**}**

**int main(void) /\* Declaring Main function name as "Main" \*/**

**{**

**Command pt;**

**int n; /\* Above we declare n as Integer number\*/**

**if (Markal(Mark, Find) == Caution) {**

**/\* Here we USE if, else condition to get error or not\*/**

**/\* Passing 2 parameters Mark and Find, if parameters is equal to Caution then it shows Markal \*/**

**perror("Markal");**

**exit(EXIT\_FAILURE); /\* if parameters equal to Caution then code resulting exit which EXIT\_FAILURE \*/**

**}**

**while (!done) {**

**/\* In this case we are going to use while loop inside if clause**

**/\* Here we while loop , Loop starts when while is not done \*/**

**char \*line;**

**/\* Initiating Variable line and it's data type is Character \*/**

**line = readline("> ");**

**/\* The characters are taking into variable line till the specified symbol mentioned aove \*/**

**if (!line) {**

**/\* Here we are using IF clause and clause starts when it is not line or we can say no line \*/**

**/\* When the cluase starts if there no line ,then it is Encountered EOF at top level \*/**

**done = 1;**

**/\* In end of the clause it shows 1 \*/**

**}**

**else {**

**/\***

**\* Remove leading and trailing whitespace from the line**

**\* Then, if there is anything left, add it to the history list**

**\* and execute it.**

**\*/**

**Belt(line);**

**/\* if above if clause not satisfied the it will print else condition prints line \*/**

**if (\*line) {**

**Past(line);**

**/\* let's execute the line\*/**

**n = parse(line, &pt);**

**Initiate(pt);**

**}**

**}**

**if(line) {**

**/\* Initiating If clause and taking "line" parameter \*/**

**free(line);**

**}**

**}**

**return 0;**

**}**

**/\* Name of the function is "Initiate" \*/**

**/\* overall descripition is Run the command \*/**

**/\* arguments used in the below function is "pt" \*/**

**void**

**Initiate(Command pt)**

**{**

**ID\_t ID;**

**/\* Similarly `cd' also must be shell-builtins. \*/**

**if (strcmp(pt.NNN->NNNlist[0], "exit") == 0) { done = 1; }**

**/\* `exit' must be in shell- builtins\*/**

**if (strcmp(pt.NNN->NNNlist[0], "cd") == 0) {**

**/\* Similarly `cd' also must be shell-builtins. \*/**

**if (chdir(pt.NNN->NNNlist[1] == NULL**

**? getenv("HOME")**

**: pt.NNN->NNNlist[1]) < 0) {**

**perror("cd");**

**}**

**return;**

**}**

**/\* First we created the entire pipeline with all the required ones \*/**

**/\* Now I run the entire pipeline as a single child process \*/**

**/\* It is easy to run entire pipeline at once to reduce " Complexity " \*/**

**/\* For all the n-commands there is n-1 forks \*/**

**/\* which then forks n-1 times for n commands (n-1 grandchildren) \*/**

**/\* leaving the child of this process to run the final command. \*/**

**\*/**

**if ( (ID = fork()) == 0) {**

**Run(pt.NNN, pt, 0, -1);**

**exit(EXIT\_FAILURE);**

**}**

**else {**

**/\* Now we are need not to wait for background commands \*/**

**if (!pt.bakground) {**

**waitID(ID, NULL, 0);**

**}**

**}**

**return;**

**}**

**/\* Name of the below mentioned function is "Run" \*/**

**/\* The overall description is an auxilary function to " Intiate function which we mentioned above \*/**

**/\* We use couple of arguments in the function " Run" , those are - p, pt,ptNum, Pipelinewrite \*/**

**/\* p means = The pipeline.\*/**

**/\* pt means = Command including redirections and backgrounding.\*/**

**/\* ptNum means = Command number Initiateing from last = 0, penultimate = 1, etc.\*/**

**/\* Pipelinewrite means = The end of the pipe to communicate with the next command.\*/**

**void**

**Run(NNN \*p, Command pt, int ptNum, int Pipelinewrite)**

**{**

**ID\_t ID;**

**char \*\*program;**

**int fd[2];**

**int Firstpt;**

**int Lastpt;**

**int stdinRedirect;**

**int stdinFile;**

**int stdoutRedirect;**

**int stdoutFile;**

**Firstpt = p->next == NULL;**

**Lastpt =ptNum == 0;**

**stdinRedirect = pt.rstdin && Firstpt;**

**stdoutRedirect = pt.rstdout && Lastpt;**

**stdinFile = -1;**

**stdoutFile = -1;**

**program = p->NNNlist;**

**/\* I am not creating any new pipe line here , I thnk for we dont need a new pipe line for the first command \*/**

**if (!Firstpt) {**

**/\* In if clause we are taking parameter "Firstpt" and this if clause executes when there is no Firstpt or not Firstpt \*/**

**if (pipe(fd) < 0)**

**/\* Here we initialize a condition only values which are less than zero \*/**

**{ perror("pipe"); exit(EXIT\_FAILURE); }**

**Run(p->next, pt,ptNum + 1, fd[1]);**

**}**

**/\* In below we are replacing one with another in my first command \*/**

**/\* Now we are replacing stdin in with Redirect standard in my first command with a file \*/**

**/\* from here stdin is exist \*/**

**if (stdinRedirect) {**

**if ( (stdinFile = open(pt.rstdin, O\_RDONLY)) < 0)**

**{ perror("open"); exit(EXIT\_FAILURE); }**

**if (dup2(stdinFile, 0) < 0)**

**{ perror("dup2"); exit(EXIT\_FAILURE); }**

**}**

**/\* Redirect standard output by replacing STDOUT of last command with a file \*/**

**/\* In below we are replacing one with another in my first command \*/**

**/\* Now we are replacing stdout with Redirect standard output my first command with a file \*/**

**/\* from here stdout is exist \*/**

**if (stdoutRedirect) {**

**if ( (stdoutFile = open(pt.rstdout, O\_WRONLY|O\_CREAT, 0666)) < 0)**

**{ perror("open"); exit(EXIT\_FAILURE); }**

**if (dup2(stdoutFile, 1) < 0)**

**{ perror("dup2"); exit(EXIT\_FAILURE); }**

**}**

**/\* Here I think the Last command is not forked \*/**

**if (Lastpt) {**

**/\* Here we are using More than a single command! \*/**

**if (!Firstpt) {**

**if (dup2(fd[0], 0) < 0)**

**{ perror("dup2"); exit(EXIT\_FAILURE); }**

**}**

**execvp(program[0], program);**

**fprintf(stderr, "Failure running `%s'.\n", program[0]);**

**exit(EXIT\_FAILURE);**

**}**

**if ( (ID = fork()) == 0) {**

**if (Firstpt) {**

**if (dup2(Pipelinewrite, 1) < 0)**

**{ perror("dup2"); exit(EXIT\_FAILURE); }**

**execvp(program[0], program);**

**fprintf(stderr, "Failure running `%s'.\n", program[0]);**

**exit(EXIT\_FAILURE);**

**}**

**/\* we always make sure , placing commands in the middle) \*/**

**else {**

**if (dup2(fd[0], 0) < 0)**

**{ perror("dup2"); exit(EXIT\_FAILURE); }**

**if (dup2(Pipelinewrite, 1) < 0)**

**{ perror("dup2"); exit(EXIT\_FAILURE); }**

**execvp(program[0], program);**

**fprintf(stderr, "Failure running `%s'.\n", program[0]);**

**exit(EXIT\_FAILURE);**

**}**

**}**

**/\* Here we are cleaning all mess or if anything leftout \*/**

**if (stdinRedirect) {**

**if (close(stdinFile) < 0)**

**{ perror("close"); exit(EXIT\_FAILURE); }**

**}**

**if (stdoutRedirect) {**

**if (close(stdoutFile) < 0)**

**{ perror("close"); exit(EXIT\_FAILURE); }**

**}**

**if (close(Pipelinewrite) < 0)**

**{ perror("close"), exit(EXIT\_FAILURE); }**

**return;**

**}**

**/\* Nmae of the function we use below is : Belt \*/**

**/\* Overall Description about the fuction is Belt whitespace from the Indicate and end end of string \*/**

**void**

**Belt (char \*string)**

**{**

**register int i = 0;**

**while (whitespace( string[i] )) {**

**i++;**

**}**

**if (i) {**

**strcpy (string, string + i);**

**}**

**i = strlen( string ) - 1;**

**while (i> 0 && whitespace (string[i])) {**

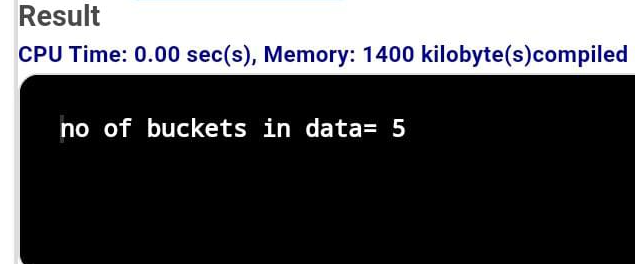
**i--;**

**}**

**string [++i] = '\0';**

**}**

**Output:**

****