**Assignment 5**

**T Karthikeyan**

**CED18I064**

**All my codes can be found here** [**CLICK HERE**](https://drive.google.com/drive/folders/1sR9-BIAcAbrUzVn_ct5n9VljeRjtqD5Z?usp=sharing)

**(1) Parent sets up a string which is read by child, reversed there and read back the parent**

**CODE**

**#include<string.h>**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h> //linux specific library for fork related calls**

**#include<time.h>**

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

**#include<errno.h>**

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

**{**

**int fd[2];**

**int fd2[2];**

**//fd[0] for read**

**//fd[1] for write**

**pipe(fd);**

**pipe(fd2);**

**int id = fork();**

**if(id == -1)**

**{**

**printf("fork error\n");**

**return 1;**

**}**

**else if(id > 0) // parent**

**{**

**char str[108] = "Operating systemLab Assignment 5";**

**char reverstr[108];**

**int slen = strlen(str);**

**close(fd[0]);**

**write(fd[1], &slen, sizeof(slen));**

**write(fd[1], str, strlen(str)\*sizeof(char));**

**close(fd[1]);**

**printf("Sent string from parent = %s\n", str);**

**close(fd2[1]);**

**read(fd2[0], reverstr, slen\*sizeof(char));**

**close(fd2[0]);**

**printf("Received string from child = %s\n", reverstr);**

**wait(NULL);**

**}**

**else // child**

**{**

**char rstr[108];**

**char revstr[108];**

**int rslen;**

**close(fd[1]);**

**read(fd[0], &rslen, sizeof(rslen));**

**read(fd[0], rstr, rslen);**

**close(fd[0]);**

**for(int k = rslen-1, j = 0; k >= 0; k--, j++)**

**revstr[j] = rstr[k];**

**close(fd2[0]);**

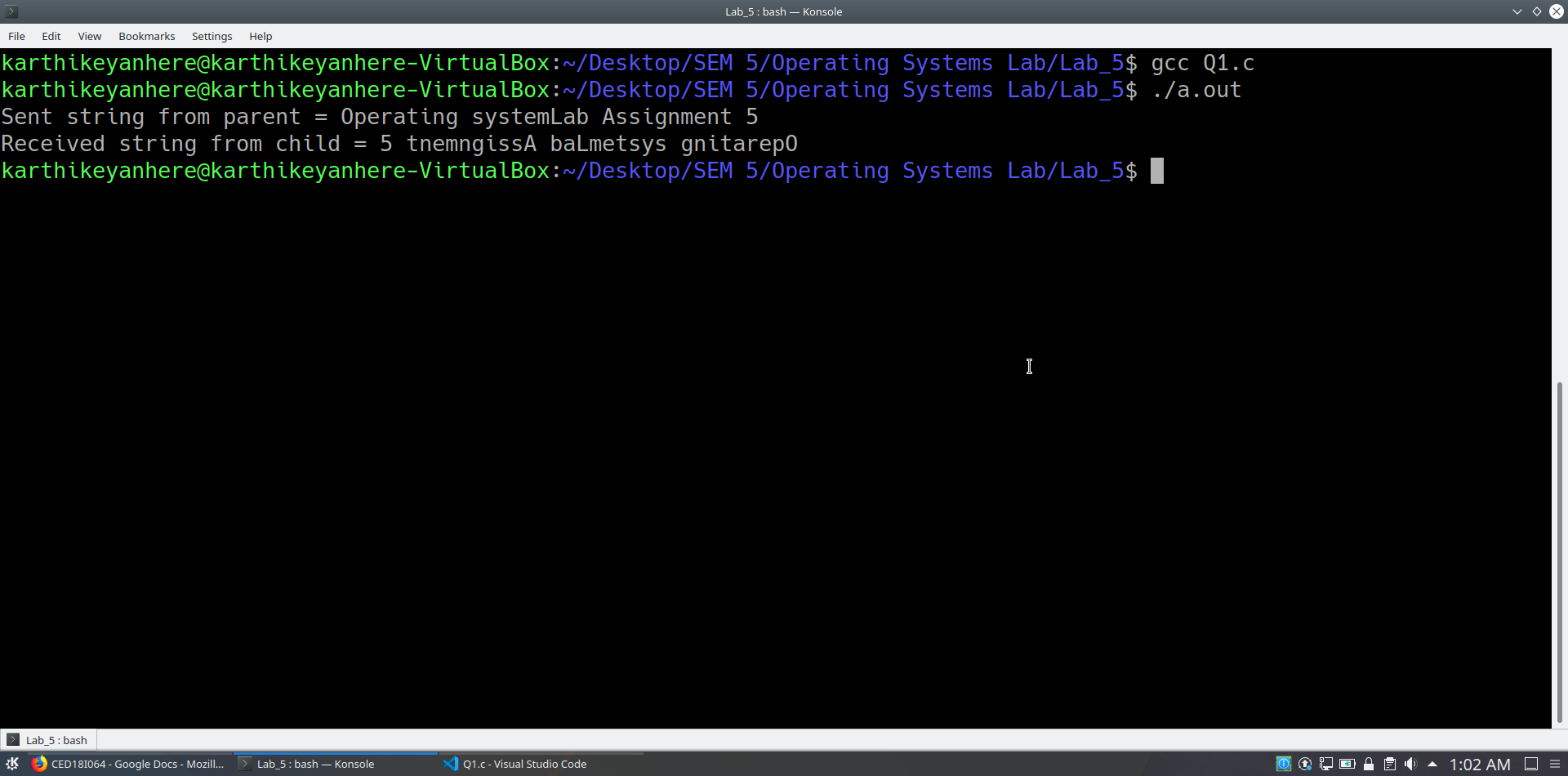
**write(fd2[1], revstr, rslen\*sizeof(char));**

**close(fd2[1]);**

**}**

**return 0;**

**}**

**OUTPUT**

**(2) Parent sets up string 1 and child sets up string 2. String 2 concatenated to string 1 at parent end and then read back at the child end.**

**CODE**

**#include<string.h>**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h> //linux specific library for fork related calls**

**#include<time.h>**

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

**#include<errno.h>**

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

**{**

**int fd1[2];**

**int fd2[2];**

**//fd[0] for read**

**//fd[1] for write**

**pipe(fd1);**

**pipe(fd2);**

**int id = fork();**

**if(id == -1)**

**{**

**printf("fork error\n");**

**return 1;**

**}**

**else if(id > 0) // parent**

**{**

**char str1[108] = "Operating systems course taught by ";**

**printf("String 1 generated by parent is #%s#\n", str1);**

**char rstr[108];**

**int s1len = strlen(str1);**

**int rslen;**

**close(fd1[1]);**

**read(fd1[0], &rslen, sizeof(int));**

**read(fd1[0], rstr, rslen\*sizeof(char));**

**close(fd1[0]);**

**strcat(str1, rstr);**

**printf("Final string from child = %s\n", str1);**

**s1len = strlen(str1);**

**close(fd2[0]);**

**write(fd2[1], &s1len, sizeof(int));**

**write(fd2[1], str1, s1len\*sizeof(char));**

**close(fd2[1]);**

**wait(NULL);**

**}**

**else // child**

**{**

**char str2[108] = "Sivaselvan sir";**

**printf("String 2 generated by child is #%s#\n", str2);**

**int s2len = strlen(str2);**

**int slen;**

**close(fd1[0]);**

**write(fd1[1], &s2len, sizeof(int));**

**write(fd1[1], str2, s2len\*sizeof(char));**

**close(fd1[1]);**

**close(fd2[1]);**

**read(fd2[0], &slen, sizeof(int));**

**read(fd2[0], str2, slen\*sizeof(char));**

**close(fd2[0]);**

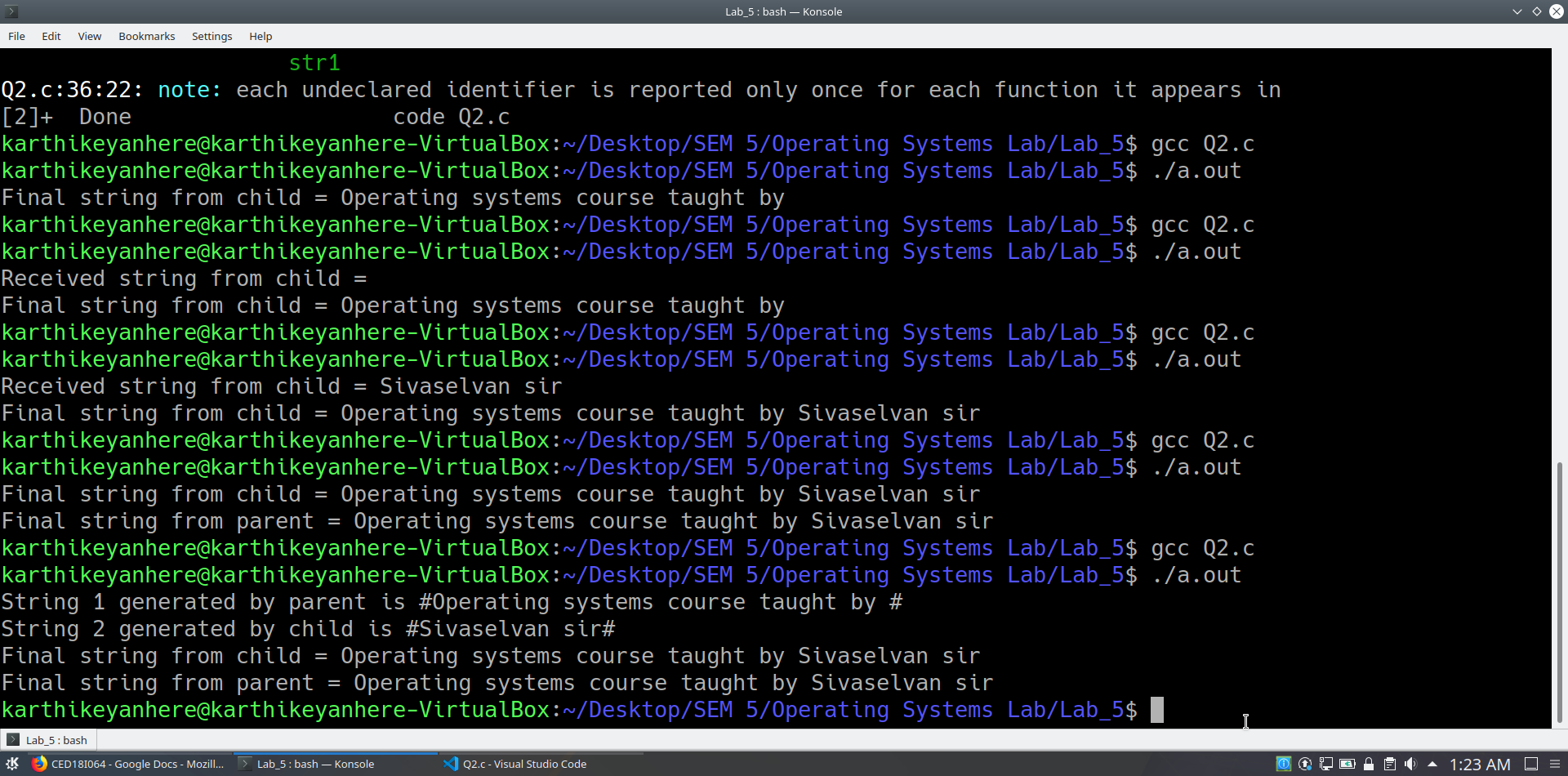
**printf("Final string from parent = %s\n", str2);**

**}**

**return 0;**

**}**

**OUTPUT**

****

**(3) Substring generation at the child end of a string setup at the parent process end.**

**CODE**

**#include<string.h>**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h> //linux specific library for fork related calls**

**#include<time.h>**

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

**#include<errno.h>**

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

**{**

**int fd1[2];**

**//fd[0] for read**

**//fd[1] for write**

**pipe(fd1);**

**int id = fork();**

**if(id == -1)**

**{**

**printf("fork error\n");**

**return 1;**

**}**

**else if(id > 0) // parent**

**{**

**char str[108] = "OperatingSystemCourse";**

**printf("String generated by parent = %s\n", str);**

**int slen = strlen(str);**

**int start, end;**

**printf("Enter the start index of substring : ");**

**scanf("%d", &start);**

**printf("Enter the end index of substring : ");**

**scanf("%d", &end);**

**close(fd1[0]);**

**write(fd1[1], &slen, sizeof(int));**

**write(fd1[1], str, strlen(str)\*sizeof(char));**

**write(fd1[1], &start, sizeof(int));**

**write(fd1[1], &end, sizeof(int));**

**close(fd1[1]);**

**wait(NULL);**

**}**

**else // child**

**{**

**char rstr[108];**

**int rslen;**

**int start, end;**

**char substr[108];**

**close(fd1[1]);**

**read(fd1[0], &rslen, sizeof(int));**

**read(fd1[0], rstr, rslen\*sizeof(char));**

**read(fd1[0], &start, sizeof(int));**

**read(fd1[0], &end, sizeof(int));**

**close(fd1[0]);**

**for(int i = start; i <= end; i++)**

**substr[i-start] = rstr[i];**

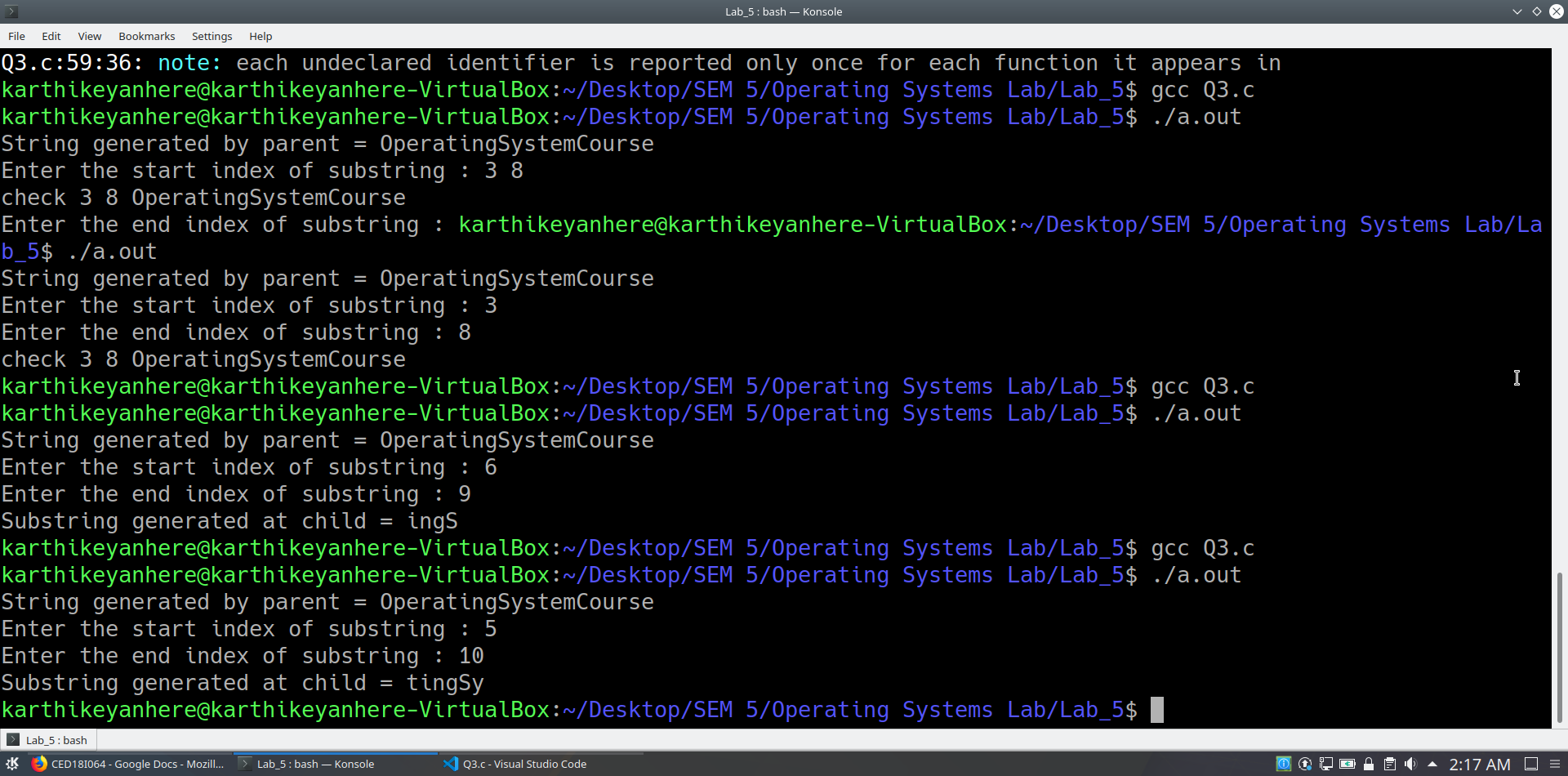
**printf("Substring generated at child = %s\n", substr);**

**}**

**return 0;**

**}**

**OUTPUT**

****

**(4) String reversal and palindrome check using pipes / shared memory.**

**CODE**

**#include<string.h>**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<unistd.h> //linux specific library for fork related calls**

**#include<time.h>**

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

**#include<errno.h>**

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

**{**

**int fd[2];**

**int fd2[2];**

**//fd[0] for read**

**//fd[1] for write**

**pipe(fd);**

**pipe(fd2);**

**int id = fork();**

**if(id == -1)**

**{**

**printf("fork error\n");**

**return 1;**

**}**

**else if(id > 0) // parent**

**{**

**char str[108] = "MalayalaM";**

**char reverstr[108];**

**int slen = strlen(str);**

**close(fd[0]);**

**write(fd[1], &slen, sizeof(slen));**

**write(fd[1], str, strlen(str)\*sizeof(char));**

**close(fd[1]);**

**printf("<inside parent>Sent string from parent = %s\n", str);**

**close(fd2[1]);**

**read(fd2[0], reverstr, slen\*sizeof(char));**

**close(fd2[0]);**

**printf("<inside parent>Received string from child = %s\n", reverstr);**

**printf("<inside parent>Pallindrome checking ...\n");**

**if(strcmp(str,reverstr) == 0)**

**printf("<inside parent>String is pallindrome\n");**

**else**

**printf("<inside parent>String is not a pallindrome\n");**

**wait(NULL);**

**}**

**else // child**

**{**

**char rstr[108];**

**char revstr[108];**

**int rslen;**

**close(fd[1]);**

**read(fd[0], &rslen, sizeof(rslen));**

**read(fd[0], rstr, rslen);**

**close(fd[0]);**

**printf("<inside child>String reversing ...\n");**

**for(int k = rslen-1, j = 0; k >= 0; k--, j++)**

**revstr[j] = rstr[k];**

**close(fd2[0]);**

**write(fd2[1], revstr, rslen\*sizeof(char));**

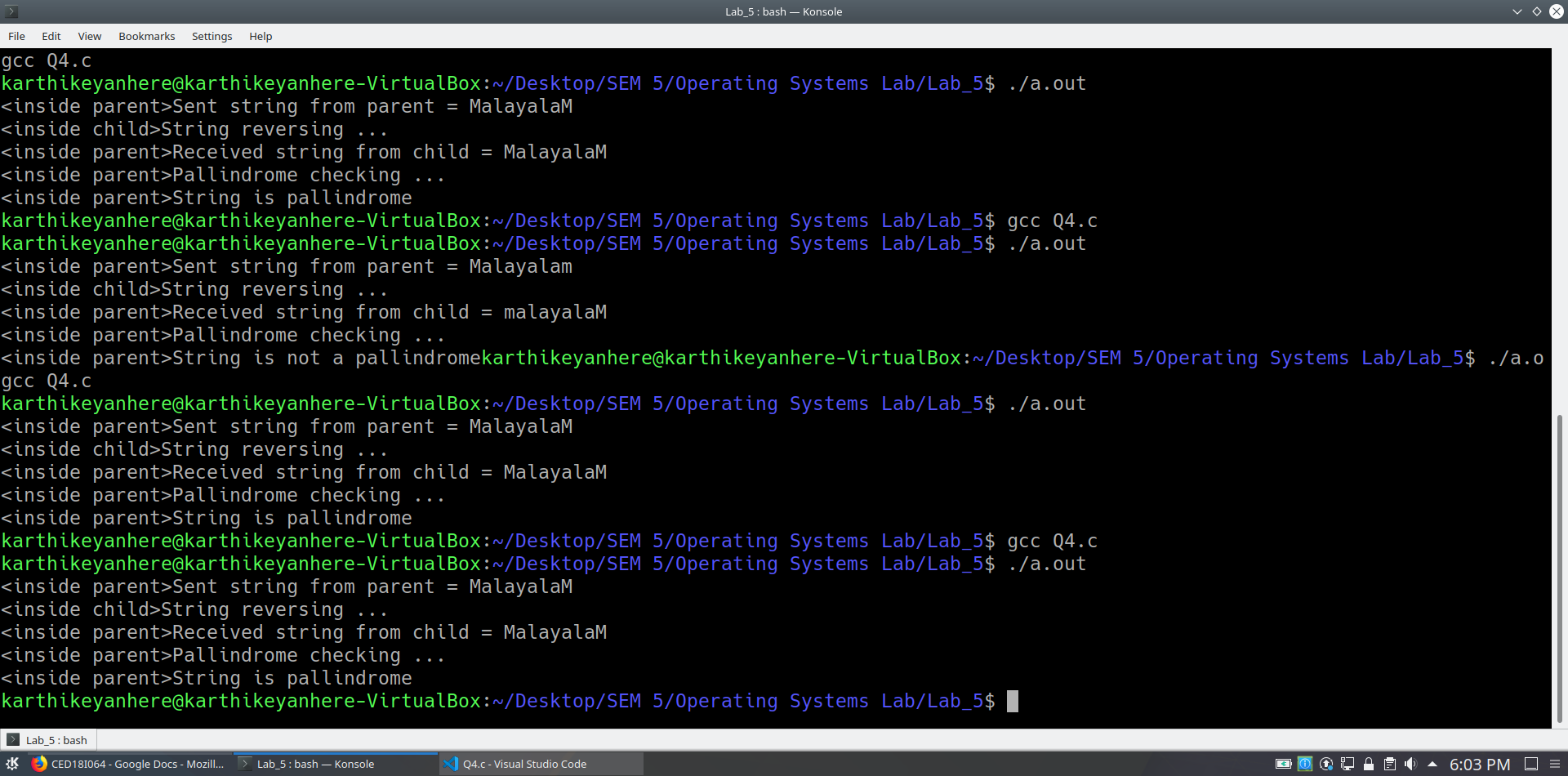
**close(fd2[1]);**

**}**

**return 0;**

**}**

**OUTPUT**

****

**(5) Armstrong number generation within a range. The digit extraction, cubing can be the responsibility of the child while the checking for sum == no can happen in the child and the output list in the child.**

**(with pipes)**

**CODE**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <sys/types.h>**

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

**#include <math.h>**

**long long ArmStrCalculation(int n)**

**{**

**int te = n;**

**int ND = 0;**

**while(te!=0) // number of digits in number**

**{**

**ND++;**

**te/=10;**

**}**

**long long sum = 0;**

**while(n!=0) // armstrong number calculation**

**{**

**int temp = n%10;**

**sum = sum + pow(temp,ND);**

**n/=10;**

**}**

**return sum;**

**}**

**int main()**

**{**

**int fd[2];**

**int fd2[2];**

**pipe(fd2);**

**//fd[0] - read**

**//fd[1] - write**

**if(pipe(fd)==-1)**

**printf("Pipe error\n");**

**int a,b;**

**printf("Enter the range [a, b] : ");**

**scanf("%d %d",&a, &b);**

**int pid;**

**pid = fork();**

**if(pid > 0) // parent**

**{**

**long long Result[b-a+1];**

**long long FinRes[b-a+1];**

**int fr = 0;**

**close(fd[1]);**

**read(fd[0], Result, (b-a+1)\*sizeof(long long));**

**close(fd[0]);**

**printf("<inside parent>Separating Armstrong numbers from rest ...\n");**

**for(int i = 0; i < (b-a+1); i++)**

**{**

**if(Result[i] == (a+i))**

**{**

**FinRes[fr] = Result[i];**

**fr++;**

**}**

**}**

**printf("<inside parent>Sending Armstrong numbers to child ...\n");**

**close(fd2[0]);**

**write(fd2[1], &fr, sizeof(int));**

**write(fd2[1], FinRes, fr\*sizeof(long long));**

**close(fd2[1]);**

**}**

**else if(pid == 0) // child**

**{**

**long long sum[b-a+1];**

**long long Finres[b-a+1];**

**int fr;**

**close(fd[0]);**

**printf("<inside child>Digit extraction and power calculation ...\n");**

**for(int i = 0; i < (b-a+1); i++)**

**{**

**sum[i] = ArmStrCalculation(a+i);**

**}**

**write(fd[1], sum, (b-a+1)\*sizeof(long long));**

**close(fd[1]);**

**close(fd2[1]);**

**read(fd2[0], &fr, sizeof(int));**

**read(fd2[0], Finres, fr\*sizeof(long long));**

**close(fd2[0]);**

**printf("<inside child>List of Armstrong numbers : ");**

**for(int k = 0; k < fr; k++)**

**{**

**printf("%lld ", Finres[k]);**

**}**

**printf("\n");**

**}**

**else**

**{**

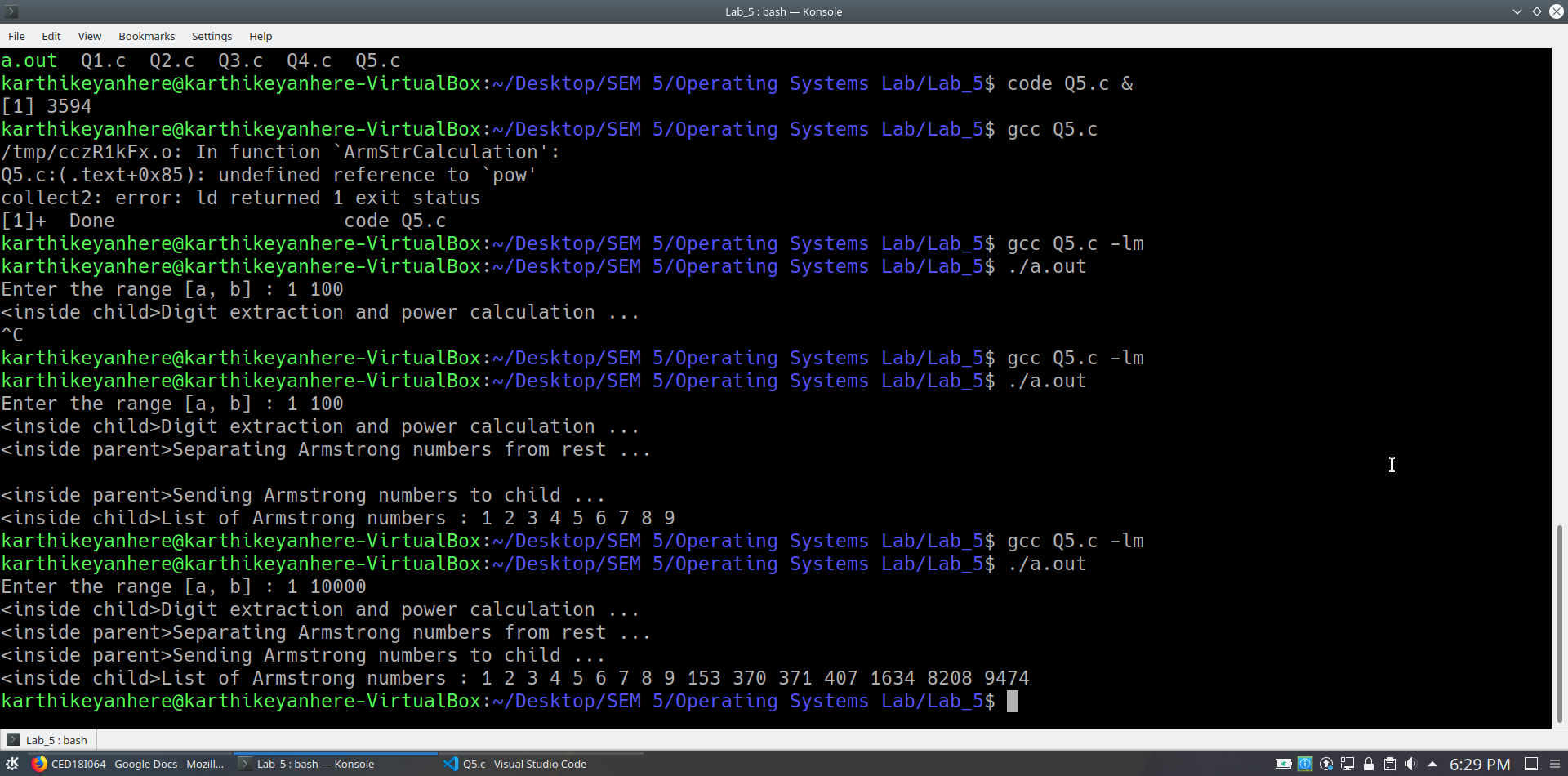
**printf("Fork failed\n");**

**}**

**return 0;**

**}**

**OUTPUT**

****

**(with pthreads)**

**CODE**

**#include <pthread.h>**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <unistd.h>**

**#include <string.h>**

**#include <sys/types.h>**

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

**#include <math.h>**

**int arr[100000];**

**long long rrr[100000];**

**long long ArmStrCalculation(int n)**

**{**

**int te = n;**

**int ND = 0;**

**while(te!=0) // number of digits in number**

**{**

**ND++;**

**te/=10;**

**}**

**long long sum = 0;**

**while(n!=0) // armstrong number calculation**

**{**

**int temp = n%10;**

**sum = sum + pow(temp,ND);**

**n/=10;**

**}**

**return sum;**

**}**

**struct block**

**{**

**int x;**

**int y;**

**};**

**void \*runner(void \*param)**

**{**

**struct block \*data = param;**

**for(int y = data->x; y <= data->y; y++)**

**{**

**rrr[y] = ArmStrCalculation(arr[y]);**

**}**

**pthread\_exit(0);**

**}**

**int main()**

**{**

**int a,b;**

**printf("Enter the range [a, b] : ");**

**scanf("%d",&a);**

**scanf("%d",&b);**

**for(int k = a; k <= b; k++)**

**arr[k] = k;**

**int start[4];**

**int end[4];**

**int n = b - a + 1;**

**int mid1 = a + (n/4);**

**int mid2 = a + 2\*(n/4);**

**int mid3 = a + 3\*(n/4);**

**start[0] = a; end[0] = mid1 - 1;**

**start[1] = mid1; end[1] = mid2 - 1;**

**start[2] = mid2; end[2] = mid3 - 1;**

**start[3] = mid3; end[3] = b;**

**pthread\_t tid;**

**pthread\_attr\_t attr;**

**for(int z = 0; z < 4; z++)**

**{**

**struct block \*data = (struct block \*)malloc(sizeof(struct block));**

**data->x = start[z];**

**data->y = end[z];**

**pthread\_t tid;**

**pthread\_attr\_t attr;**

**pthread\_attr\_init(&attr);**

**pthread\_create(&tid, &attr, runner, data);**

**pthread\_join(tid, NULL);**

**}**

**printf("List of Armstrong numbers : ");**

**for(int i = a; i <= b; i++)**

**{**

**if(arr[i] == rrr[i])**

**printf("%d ", arr[i]);**

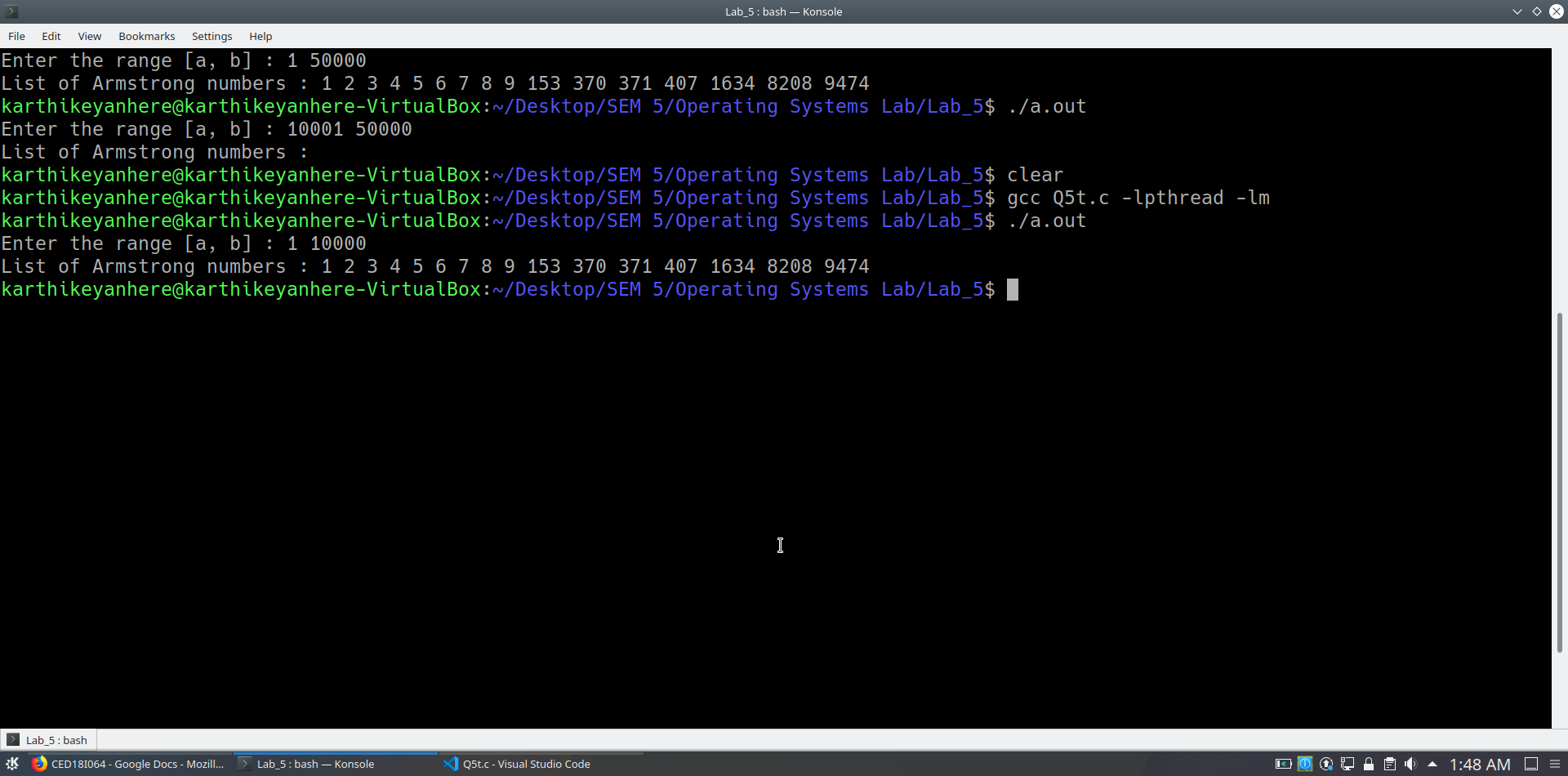
**}**

**printf("\n");**

**return 0;**

**}**

**OUTPUT**

****