Curtin University – Discipline of Computing

Declaration of Originality

Complete this form for every **online assessment** and submit as per instructions from your lecturer or unit coordinator

Last name:	Ching	Student ID:	22013213
Other name(s):	Mao Jin		
Unit name:	Unix System Programming	Unit Code:	COMP2002
Lecturer:	Foad Motalebi	Tutorial Session	-
Date of assess:	21/4/2025	Which assessment?	1

Tick all the boxes below to indicate that you agree with the following:

I declare that:

- ▣ The above information is complete and accurate.
- ▣ The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- ▣ I have not received assistance from anyone else during the online assessment.
- ▣ I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.

I understand that:

- ▣ Plagiarism and collusion are dishonest, and unfair to all other students.
- ▣ Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- ▣ I am aware that investigation of academic misconduct may require me to sit an interview should it be needed.
- ▣ If I plagiarize or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- ▣ Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- ▣ It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: CMJ

Date of signature: 4/24/25

Introduction

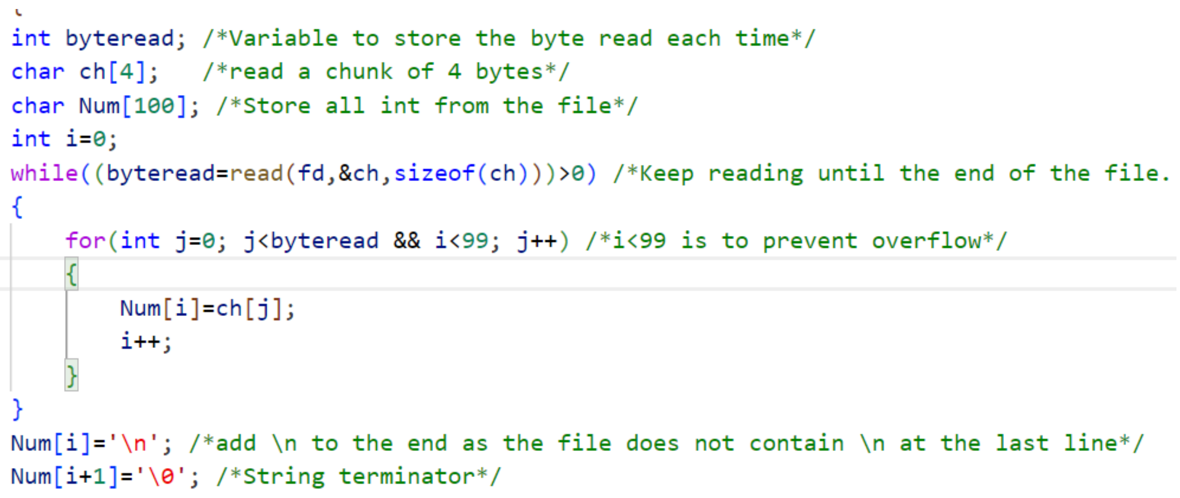
This assignment is about creating child processes and each child process will check primality for each integer from the file. The parent is responsible for writing each integer to the child and child has to pass the integer to isPrime function and return the result to parent. I include a flowchart of child and parent process under the challenges section. This short report documents how my implementation resolves the following issues. I will take a screenshot of my code snippet to ensure that my implementation is aligned with my explanation. Besides, I will include a walkthrough of my code to better understand the logic.

Short Report

Q1. How does your program read() and parse the integers correctly from the text file and store them into 1D array?

Ans

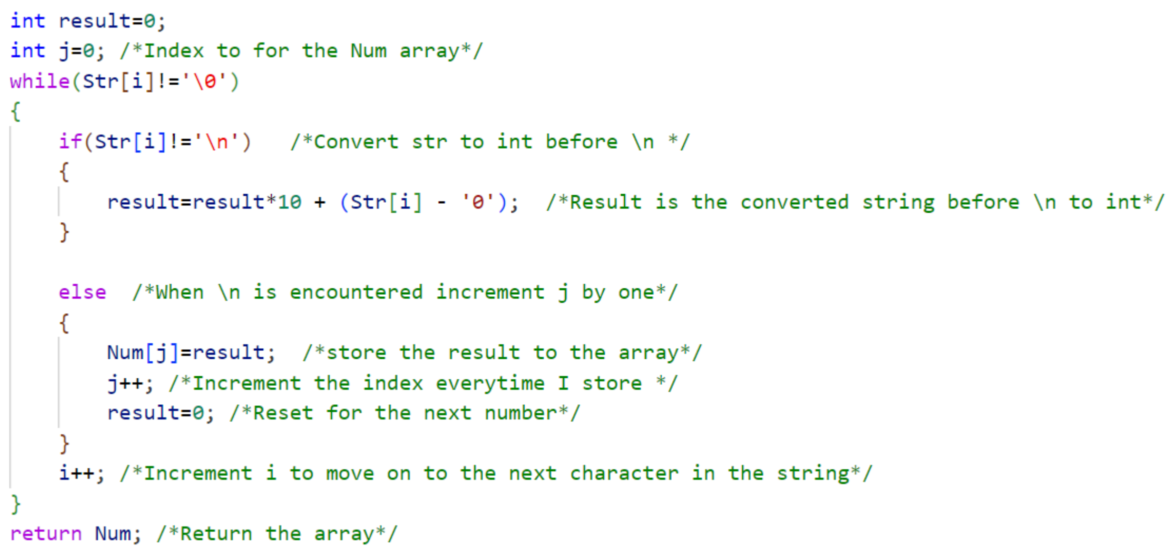
Reading: My program will read up to 4 characters at a time until the end of the file. Each character read will be stored in an array called Num in my main function. Each integer in the file is seperated by new line which will be read as \n by read() function and this character is also stored in the array. However, the last line does not include \n which mean I need to manually add \n for parsing the integers and a string terminator \0 after read.



```
int byteread; /*Variable to store the byte read each time*/
char ch[4];   /*read a chunk of 4 bytes*/
char Num[100]; /*Store all int from the file*/
int i=0;
while((byteread=read(fd,&ch,sizeof(ch)))>0) /*Keep reading until the end of the file.
{
    for(int j=0; j<byteread && i<99; j++) /*i<99 is to prevent overflow*/
    {
        Num[i]=ch[j];
        i++;
    }
}
Num[i]='\n'; /*add \n to the end as the file does not contain \n at the last line*/
Num[i+1]='\0'; /*String terminator*/
```

Image 1 shows how program reads the file and stores into a char array, Num, before converting to int

Parsing: The array Num now contains char array. This is passed over to the function called str_to_int to convert each character before \n to integer and this integer will be stored in N array. At the same time, the arraysize can be obtained by converting the first character up to the first occurrence of \n to integer. This integer represents the size of the array.



```
int result=0;
int j=0; /*Index to for the Num array*/
while(Str[i]!='\0')
{
    if(Str[i]!='\n')   /*Convert str to int before \n */
    {
        result=result*10 + (Str[i] - '0');  /*Result is the converted string before \n to int*/
    }

    else  /*When \n is encountered increment j by one*/
    {
        Num[j]=result;  /*store the result to the array*/
        j++; /*Increment the index everytime I store */
        result=0; /*Reset for the next number*/
    }
    i++; /*Increment i to move on to the next character in the string*/
}
return Num; /*Return the array*/
```

As you can see from the code, the program will keep converting each character to int before \n by applying the formula ,result * 10 + (Str[i] – ‘0’). Please look at the walkthrough of an example below to better understand my logic.

Eg: char *Str= “123\n…\n\0”

In 1st iteration:

Result =0

Result=0 * 10 + (‘1’ – ‘0’)=0 + 49(ascii value of 1) – 48(ascii value of 0)=1

In 2nd iteration:

Result=1

Result= 1 * 10 + (‘2’ – ‘0’)=10 + 50(ascii value of 2) – 48(ascii value of 0)=12

In 3rd iteration:

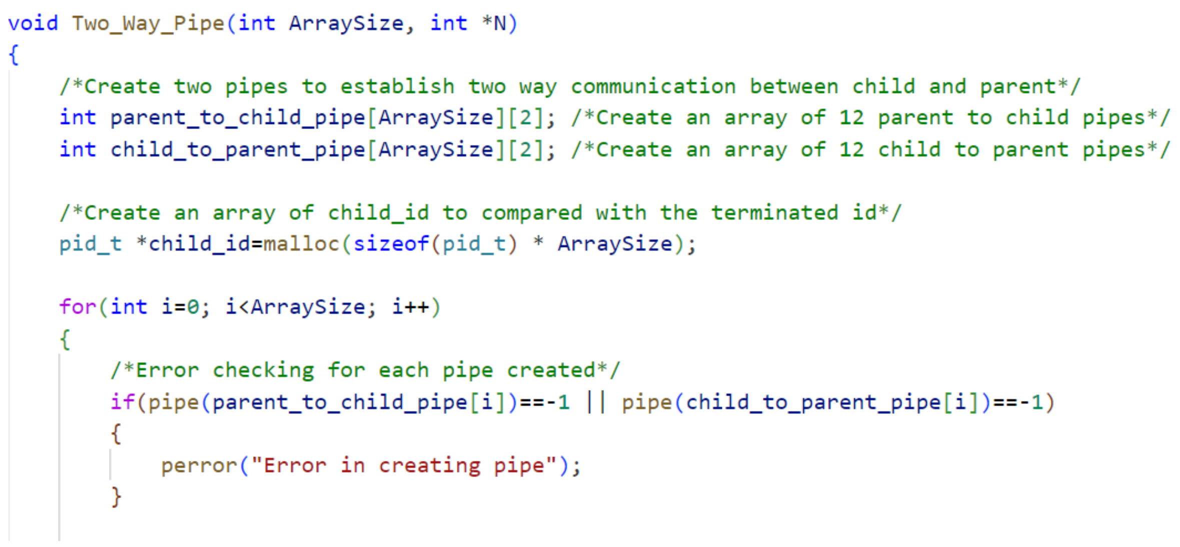
Result=12

Result= 12 * 10 + (‘3’ – ‘0’)= 120 + 51(ascii value of 3) – 48(ascii value of 0)=123

The next character is \n which causes the program to enter the else statement where the result will be stored in the Num array. This process repeats until it reaches the string terminator, \0.

Q2. How does your program create and organise the pipes? How do the parent and child processes know which pipe to use?

Ans: My program creates two pipes using pipe() during each iteration. Each pipe has two file descriptors, 0 for reading and 1 for writing. The number of iteration is based on the array size which is the first integer in the numbers.txt file. In each iteration, my program will use system call pipe to create two pipes so parent and child will have their respective pipes for communication over each integer and the result of primality with each other.



```c
void Two_Way_Pipe(int ArraySize, int *N)
{
    /*Create two pipes to establish two way communication between child and parent*/
    int parent_to_child_pipe[ArraySize][2]; /*Create an array of 12 parent to child pipes*/
    int child_to_parent_pipe[ArraySize][2]; /*Create an array of 12 child to parent pipes*/

    /*Create an array of child_id to compared with the terminated id*/
    pid_t *child_id=malloc(sizeof(pid_t) * ArraySize);

    for(int i=0; i<ArraySize; i++)
    {
        /*Error checking for each pipe created*/
        if(pipe(parent_to_child_pipe[i])==-1 || pipe(child_to_parent_pipe[i])==-1)
        {
            perror("Error in creating pipe");
        }
```

Walkthrough of my code:

For each iteration:

Starting from i=0 to 11

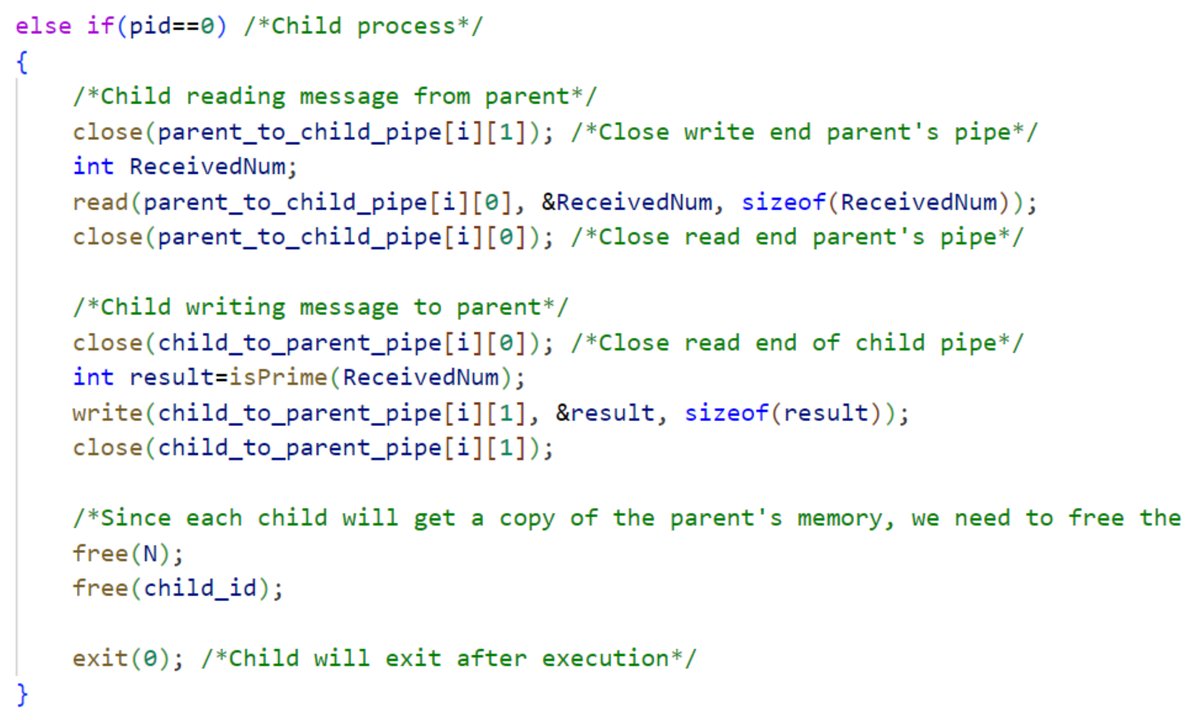
parent_to_child_pipe(P1[i]) and child_to_parent_pipe(P2[i]) are created.

Parent uses P1[i][1] to write the integer, N[i] to child.

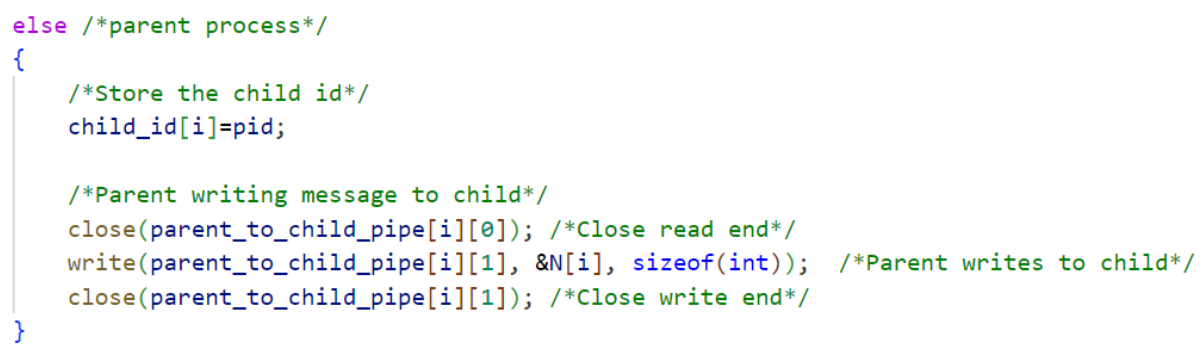
Child_id[i][0] is now able to read N[i] via P1[i][0].

Child can write the result of N[i] to parent via P2[i][1].

As you can see from the code below, each pipe is created to communicate the result and integer over each other.



```
else if(pid==0) /*Child process*/
{
    /*Child reading message from parent*/
    close(parent_to_child_pipe[i][1]); /*Close write end parent's pipe*/
    int ReceivedNum;
    read(parent_to_child_pipe[i][0], &ReceivedNum, sizeof(ReceivedNum));
    close(parent_to_child_pipe[i][0]); /*Close read end parent's pipe*/

    /*Child writing message to parent*/
    close(child_to_parent_pipe[i][0]); /*Close read end of child pipe*/
    int result=isPrime(ReceivedNum);
    write(child_to_parent_pipe[i][1], &result, sizeof(result));
    close(child_to_parent_pipe[i][1]);

    /*Since each child will get a copy of the parent's memory, we need to free the
    free(N);
    free(child_id);

    exit(0); /*Child will exit after execution*/
}
```

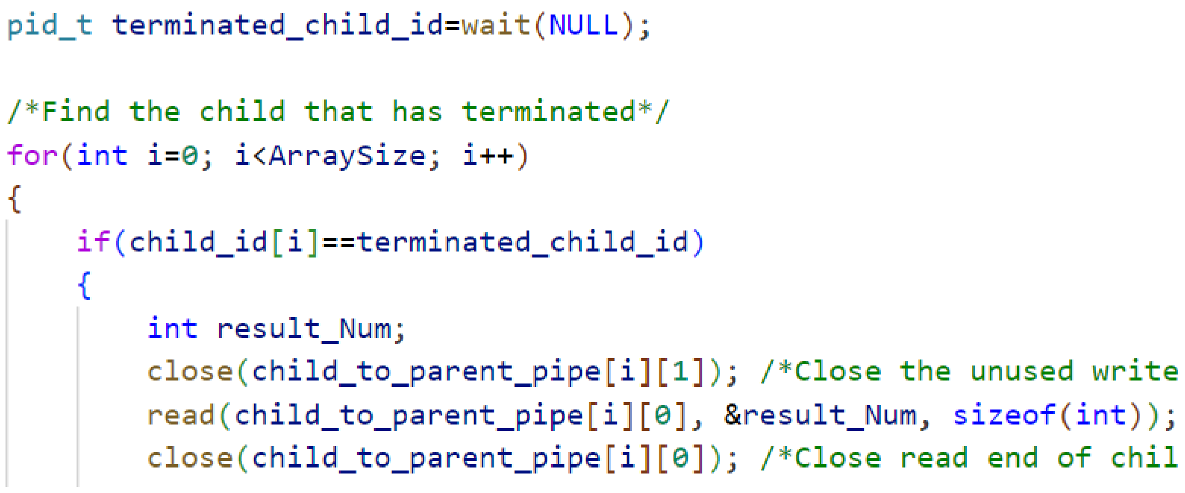


```
else /*parent process*/
{
    /*Store the child id*/
    child_id[i]=pid;

    /*Parent writing message to child*/
    close(parent_to_child_pipe[i][0]); /*Close read end*/
    write(parent_to_child_pipe[i][1], &N[i], sizeof(int));  /*Parent writes to child*/
    close(parent_to_child_pipe[i][1]); /*Close write end*/
}
```

Q3. How do the child processes know when to stop read()-ing? How do they know there is no more number coming from the parent? What does your program do to ensure the read() does not block forever?

Ans: I implemented a loop based on the array size so the child will automatically stop reading when the loop terminates. My program will always close the unused write end pipe before reading. After reading, my program will close the read end pipe. This prevents read from blocking forever. If my program does not close used pipe, it will continue waiting for read which block the other processes.

Q4. How does the parent process keep track of which child has been terminated and read() the primality test result from the correct pipe?

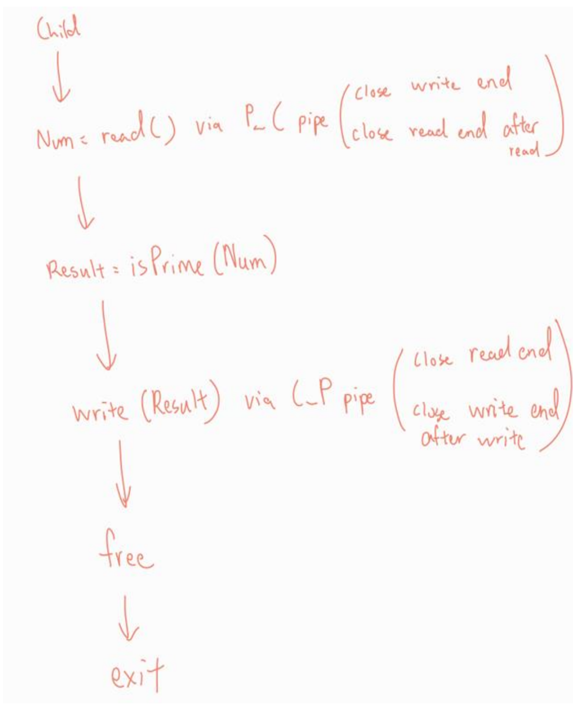


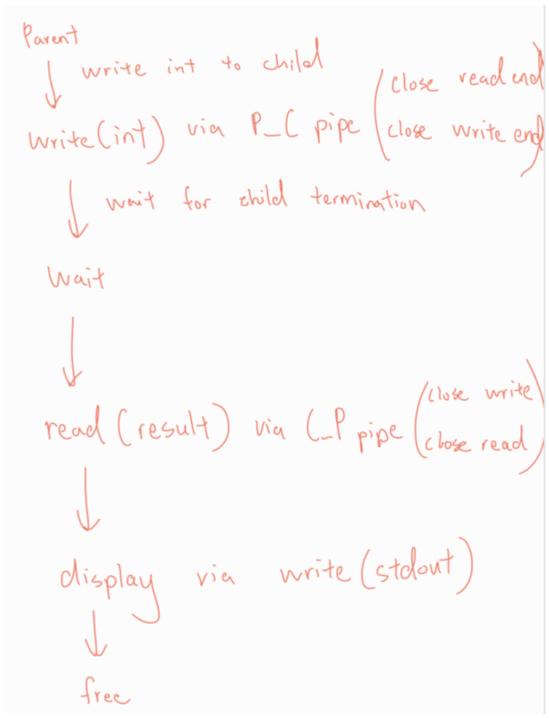
```
pid_t terminated_child_id=wait(NULL);

/*Find the child that has terminated*/
for(int i=0; i<ArraySize; i++)
{
    if(child_id[i]==terminated_child_id)
    {
        int result_Num;
        close(child_to_parent_pipe[i][1]); /*Close the unused write
        read(child_to_parent_pipe[i][0], &result_Num, sizeof(int));
        close(child_to_parent_pipe[i][0]); /*Close read end of chil
```

Ans: After forking all the child processes, the parent waits for any child to terminate before proceeding. This wait() returns the terminated child id which can then be used to compare with the child_id stored by the parent. Based on the child_id at the particular index, the parent can read the result from the pipe at the index and print to the console.

Challenges

This assignment does not allow me to use normal library functions like printf(), scanf() and all library functions related to file IO and string which increases the difficulty level. I have to use read or write system calls which requires the length of the string. This raises another problem because I cannot use strlen to calculate the length and I have to think of a way to calculate the length myself. Converting string to integer is one of the challenges that I faced. It requires me to come up with a formula that involves the use of ASCII value for the convertion of string to integer. Piping gave me a headache initially because the algorithm was very overwhelming. I came out with a flowchart to get an overview of what operation carried out by parent and child processes.


Child
Num = read() via P_C pipe
close write end
close read end after read
Result = isPrime(Num)
write(Result) via C_P pipe
close read end
close write end after write
free
exit



Reference List

GeeksforGeeks. (n.d.). *Trial division algorithm for prime factorization*. https://www.geeksforgeeks.org/trial-division-algorithm-for-prime-factorization/

Stack Overflow. (n.d.). *Posix - How to use nanosleep() in C? What are tim.tv_sec and tim.tv_nsec?* https://stackoverflow.com/questions/7684359/how-to-use-nanosleep-in-c-what-are-tim-tv-sec-and-tim-tv-nsec

University of Toronto. (n.d.). *ASCII table (7-bit) - ASCII code*. https://www.asciitable.com/