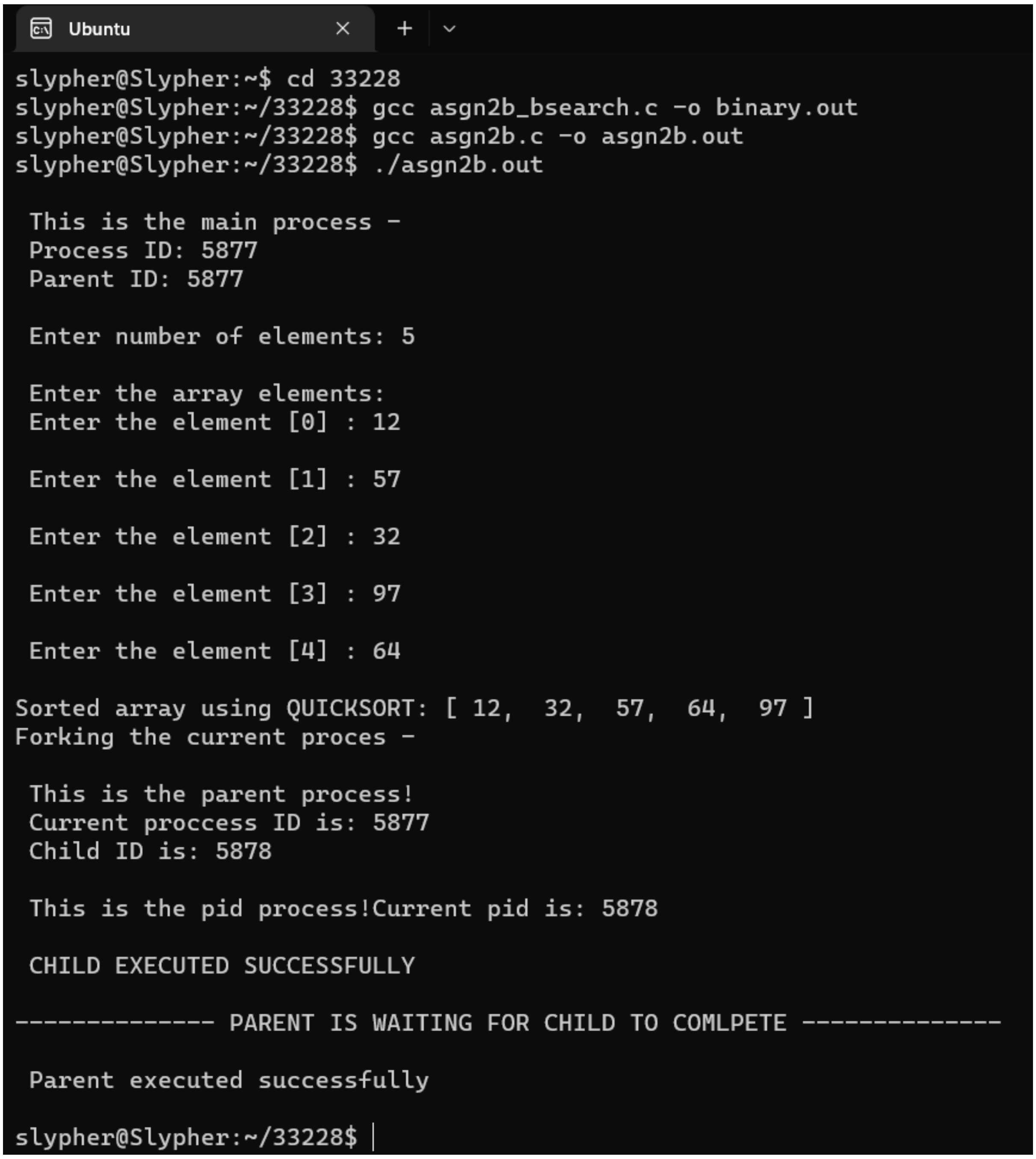
```c
/*
  Name: Ayush Gupta       Roll no: 33228
  Batch: L-10         Asgn: 2b
*/

#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/wait.h>
#include <string.h>
#define MAXSIZE 50

void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}
// function to find the partition position
int partition(int array[MAXSIZE], int low, int high) {

  // select the rightmost element as pivot
  int pivot = array[high];

  // pointer for greater element
  int i = (low - 1);

  // traverse each element of the array
  // compare them with the pivot
  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {

      // if element smaller than pivot is found
      // swap it with the greater element pointed by i
      i++;

      // swap element at i with element at j
      swap(&array[i], &array[j]);
    }
  }

  // swap the pivot element with the greater element at i
  swap(&array[i + 1], &array[high]);

  // return the partition point
  return (i + 1);
}
```

```c
void quickSort(int array[MAXSIZE], int low, int high) {
  if (low < high) {

    // find the pivot element such that
    // elements smaller than pivot are on left of pivot
    // elements greater than pivot are on right of pivot
    int pi = partition(array, low, high);

    // recursive call on the left of pivot
    quickSort(array, low, pi - 1);

    // recursive call on the right of pivot
    quickSort(array, pi + 1, high);
  }
}

// Function to print array on console
void print_arr(int arr[MAXSIZE], int n) {
    int i;
    printf("[");
    for(i = 0; i < n - 1; i++) {
        printf(" %d, ",arr[i]);
    }
    printf(" %d ]", arr[i]);
    printf("\n");
}

void main() {
    // Declare the variables
    int arr[MAXSIZE], n;
    pid_t pid;

    // The main proces starts
    printf("\n This is the main process - ");
    printf("\n Process ID: %d", getpid());
    printf("\n Parent ID: %d \n", getpid());

    // Take the array input
    printf("\n Enter number of elements: ");
    scanf("%d", &n);
    printf("\n Enter the array elements:");
    for(int i = 0; i < n; i++) {
        printf("\n Enter the element [%d] : ", i);
        scanf("%d", &arr[i]);
    }

    // Sorting the array
    printf("\nSorted array using QUICKSORT: ");
    quickSort(arr, 0, n - 1);
    print_arr(arr, n);

    // Fork System call
    printf("Forking the current proces - \n");
    pid = fork();

    if(pid == -1) {
        printf("Unfortunately the pid was not born");
    }
    else if(pid == 0) {
        // Inside the child process
        printf("\n This is the pid process!");
        printf("Current pid is: %d", getpid());

        int i;
        char *buffer[n+1];

        // arg 0 = name of executable file
        buffer[0] = "./binary";


        for(i = 1; i < n + 1; i++) {
            // Allocating memory
            buffer[i] = malloc(10);
            snprintf(buffer[i], 10, "%d", arr[i - 1]);
        }

        // Setting last element to NULL
        buffer[i] = NULL;
        // print_arr(buffer);

        // Calling pid proces on top of parent (overriding)
        execv("./binary", buffer);

        printf("\n\n CHILD EXECUTED SUCCESSFULLY \n\n");
    }
    else {
        // Inside parent process
        printf("\n This is the parent process!");
        printf("\n Current proccess ID is: %d", getpid());
        printf("\n Child ID is: %d", pid);
        printf("\n-------------- PARENT IS WAITING FOR CHILD TO COMLPETE -----
---------");
        wait(NULL);
        printf("\n\n Parent executed successfully \n\n");
    }
}
```

```c
/*
    Name: Ayush Gupta                Roll no: 33228
    Batch: L-10                  Asgn: 2b
*/

// COMPILE: gcc binary_search.c -o binary
// EXECUTE: ./binary
// child.c file

#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<sys/types.h>

// Function to print array on console
void print_arr(int arr[50], int n) {
    int i;
    printf("[");
    for(i = 0; i < n - 1; i++) {
        printf(" %d, ",arr[i]);
    }
    printf(" %d ]", arr[i]);
    printf("\n");
}

// Binary search
// A iterative binary search function. It returns
// location of x in given array arr[l..r] if present,
// otherwise -1

void binary_search(int arr[50], int l, int r, int x)
{
    while (l <= r) {
        int m = l + (r - l) / 2;

        // Check if x is present at mid
        if (arr[m] == x) {
            printf("\n %d is present at index %d \n", x, m);
            break;
        }

        // If x greater, ignore left half
        if (arr[m] < x)
            l = m + 1;

        // If x is smaller, ignore right half
        else
            r = m - 1;
    }

    // if we reach here, then element was
    // not present
    if(l>r)
        printf("\n %d was not found in the array.", x);
}

// Main function of the program
void main(int argc, char* argv[]) {
    int arr[argc - 1], search;

    for (int i = 1; i<argc; i++) {
        arr[i-1] = atoi(argv[i]);
    }

        print_arr(arr, argc - 1);

        printf("\n Enter the value to be searched: ");
        scanf("%d", &search);

        binary_search(arr, 0, argc-1, search);
        printf("\n In Child the current PID is: %d \n", getpid());

}
```



```
slypher@Slypher:~$ cd 33228
slypher@Slypher:~/33228$ gcc asgn2b_bsearch.c -o binary.out
slypher@Slypher:~/33228$ gcc asgn2b.c -o asgn2b.out
slypher@Slypher:~/33228$ ./asgn2b.out

 This is the main process -
 Process ID: 5877
 Parent ID: 5877

 Enter number of elements: 5

 Enter the array elements:
 Enter the element [0] : 12

 Enter the element [1] : 57

 Enter the element [2] : 32

 Enter the element [3] : 97

 Enter the element [4] : 64

Sorted array using QUICKSORT: [ 12,  32,  57,  64,  97 ]
Forking the current proces -

 This is the parent process!
 Current proccess ID is: 5877
 Child ID is: 5878

 This is the pid process!Current pid is: 5878

 CHILD EXECUTED SUCCESSFULLY

-------------- PARENT IS WAITING FOR CHILD TO COMLPETE --------------

 Parent executed successfully

slypher@Slypher:~/33228$ 
```