A1_Report - All Problems Solutions

- by Parikshit R Sarode

1) QUES1: Write a bash script which displays the Factorial of a number N on the monitor

.

Solution -

In the code file attached with this report, we can see that we are taking input from in num variable using the read command. We are using a variable called res initialized to 1 to store the result. We next use a while loop with the condition that the input number num be greater than 1. In the body of the loop we multiply res and num and decrement num. This method makes the loop multiply from num till 1 and store the result in res variable. The loop exists when the variable num becomes 1. After finishing the loop we get factorial in res variable and display it using echo command.

`Code attached with report`

2) QUES2: Given 2 arrays of integers, use C and pthread to write a parallel program to find the common elements. Assume the entire arrays are stored initially in one location and distributed to the different threads for parallel processing.

.

Solution -

Starting from main function, first we take input from the user. We take input in 2 arrays. The size of arrays are taken as ten in my program. 'for' loops are used to take input. The next section of the code, we create threads using the funcction pthread_create. Giving it parameter as thread1 and thread2 and the thread function as threadfun. Here both threads work in parallel to display common elements. The first thread takes half of first array and checks repetitions with second array. Thread1 checks elements from arr1[0] to arr1[4] with all elements with arr2[]. Similarly the thread2 checks repetitions with second array. Thread2 checks elements from arr1[5] to arr1[9] with all elements with arr2[].

While the 2 threads are executing parallely, the main thread is waiting for their executing completion. The threads are then joined to main using pthread_join function for both the threads.

`C language code attached`

3) QUES3: Write a program to find the count of entered numbers in parent process, sum of even numbers in child process 1 and sum of odd numbers in child process

.

Solution for Part a-

Starting from the main function, we take input from the user in a linked list because we do not know how many data will be inserted at run time. We create linked list nodes for every element and insert it in the linked list using the next pointer.

In the next section of the code, we create processes using the system call fork. First child process is created and the child process is used to calculate the sum of even numbers in the linked list. This is done by traversing the linked list using the head pointer and current pointer and a while loop. On every iteration of the while loop the data in the linked list is checked and if it is even i.e. %2==0 , we add it in the evensum variable and the process exits. The parent process in the meanwhile, counts the number of nodes of linked list to display. Since the sequence is First child and then parent, we make use of wait() system call in the parent process.

In the next part of code, main process creates another child process. This child process computes the odd number sum of the nodes of the linked list. This is done same as child process1 by traversing the linked list and %2!=0 condition is checked to select the odd numbers and adding them to the variable oddsum and printing it. The second child process ends and in the meanwhile the main process is waiting using wait() system call

.

Solution for Part b-

Starting from the main function, we take input from the user in a linked list because we do not know how many data will be inserted at run time. We create linked list nodes for every element and insert it in the linked list using the next pointer

Here we create child process 1 to compute evensum as done in part a. We create the second child in the else section of the child 1 process after completing the execution of main process. The second child the computes the oddsum from the linked list and prints it. The second process goes in sleep with sleep() system call, to let the first process finnish execution. Meanwhile the main process is waiting for the child process to execute and also sleeps to prevent further code getting ececuted in the middle of child processes. The main process then computes the count and prints the program ends.

C Code attached

4) QUES4:

1) Hand simulate the program. Using the table givenbelow, record the values of the variables X, result, answer, andy at the following lines in the program: 6,9,11,15,19,21. For somelines, some variables will not be defined, record that information instead of the variable's value.

Solution

Hand simulating

Line Number	X	Result	Answer	Y
6	5	Not defined	Not defined	Not defined
9	10	garbage	Not defined	
11	10	35	0	
15	5	Not defined	0	10
19	20	Not defined	15	10
21	20	Not defined	35	10

2) Using gdb and the break, step, and print variable commands record the values of the variables X, result, answer, and y at the following lines in the program: 6,9,11,15,19,21 in a different table. For some of these lines, you will get an error because the variable doesn't exist within the scope of that line. Do not go back and change the values in your first table

Solution

Using debugger

Line Number	X	Result	Answer	Y
6	0	0	Not shown	Not shown
9	10	0		
11	10	35	Not shown	
15	0	Not	0	10
19	20	shown	15	10
21	20		35	10

3)

Solution -

.

1) Showed 0 always.

2)line 9- 10, line 15 - 0 are the values of x

Reason - Scope changes.

3)Mistakes were make by not keeping scope in mind. The garbage value was displayed 0 .