

Write a LINUX program to do binary fan-in to add up a list of numbers. For trial purposes, we will use the numbers 1-128, however in a real application, these would be array indices to a shared array rather than the actual numbers.

It will be necessary to create two separate programs. The actual work will be done by the client process which will work as follows.

1. Read a message from anybody of type 100.
2. Extract an array of three integers from the message.
 - a. The first number is the parent process id.
 - b. The second number is the smallest number to be added. Call this First.
 - c. The third number is the largest number to be added. Call this Last.
3. If the largest and smallest numbers to be added are the same, then return the largest number to the parent in a message of type 200.
Create a message containing a single integer and send it to the parent.
4. If the largest and smallest numbers are different, spawn two client processes.
5. Break the list of numbers into two lists. Compute a new point, Middle and Middle+1.
 $Middle = (First + Last) / 2;$
6. Create a message for the first process containing an array of three integers.
 - a. The first integer is the process ID of the current process.
 - b. The second integer is First from step 2.
 - c. The third integer is Middle from step 5.
7. Send this message (type 100) to the first client you spawned in step 4.
8. Create a message for the second process containing an array of three integers.
 - a. The first integer is the process ID of the current process.
 - b. The second integer is Middle+1 from step 5.
 - c. The third integer is Last from step 2.
9. Send this message (type 100) to the second process you spawned.
10. Receive a message of type 200 from each of the spawned processes.
11. Extract a single integer from each message of type 200.
12. Add the two integers and send the result to the parent in a message of type 200.
13. Terminate the process from the pvm system.

The second program, the master program, must spawn a single client process and send a single message of type 100 to that process. Insert a 3-element array of ints into the message. The first int will be the process ID of the master process. The second will be the number 1. The third will be the number 128. The master process will then receive a single message of type 200 from the spawned process. It will extract a single integer from that message and print it with the text "The result is %d\n."

To use PVM, you need to include the following file in your programs.

`/usr/share/pvm3/include/pvm3.h`

When compiling you need to add the following argument to the command line of the cc command.

```
cc MyProg.c /usr/share/pvm3/lib/LINUXI386/libpvm3.a -o MyProg
```

When running the program you need to start the PVM daemon. To do this, type the pvm command. When you get the pvm> prompt, type “quit” and press return. When you are done using the pvm daemon, type the pvm command again, but this time type “halt” instead of “quit.”

The pvm function calls are used to interact with the pvm daemon. They will fail if the pvm daemon is not running. Here are the function calls you need. (The full description is available using the LINUX man command.)

```
int MyTid = pvm_mytid( ); // returns current process's ID
pvm_rcv(-1,100); // receive a message of type 100 from anybody
int MyArray[3];
pvm_upkint(MyArray,3,1); // extract an array of 3 integers from the most recently
                        // received message
pvm_init(pvmDataDefault); // do this to start creating a message
pvm_pkint(MyArray,3,1); // insert an array of three elements into the current
                        // message you are creating
int OneValue;
pvm_upkint(&OneValue,1,1); // extract a single integer from the most recently
                        // message
pvm_pkint(&OneValue,1,1); // insert a single integer into the message you are
                        // creating
int ParentID;
pvm_send(ParentID,200); // send a message of type 200 to the process ID contained
                        // in the variable ParentID.
int ChildID;
pvm_rcv(ChildID,200); // receive a message of type 200 from the specific process
                        // ID contained in the variable ChildID.
int ChildIDs[2];
pvm_spawn("Client",NULL,PvmTaskDefault,"",2,ChildIDs);
// spawn two new processes. The name of the program is "Client" the 2 indicates
// the number of processes to start. The "ChildIDs" array must be an array of int
// with one entry for each process spawned. The rest of the stuff is required
// constants.
pvm_exit( ); // Terminate this process from the pvm system.
```

To make this work, create the following directory in your home directory:

```
pvm3/bin/LINUXI386
```

Place all your code in this directory.