Project - 1

Design Document

Vignesh Kulothungan ([vxk111430@utdallas.edu](mailto:vxk111430@utdallas.edu))

Date: 7th February, 2014

**Algorithm:**

**Parent Process:**

* Reads the **queue limit** value from command line.
* Creates 4 pipes. 2 pipes for each child processes (2 for Factorial and 2 for Addition). One for Sending data and other to receive Acknowledgements.
* Creates two child processes using fork() command.
* Opens “instruction.dat” file. Reads each line using getsline() in a while loop. If error occurs during file read exits the loop.
* If no error during line read, it parses first 3 bytes. If it is “fac” then extracts two integers and sends the two integers to the Factorial child process.
* If the first 3 bytes are “add” then extracts a string of length 8 bytes (filename cannot be more than 8 bytes) and an integer. It sends both the filename and integer value to the Adder child process.
* Before sending data to child, parent checks the queue limit (maximum pending instructions in the queue), if queue limit is reached waits for a reply. (Queue limit is checked for 2nd cycle after sending data of 1st cycle. Since minimum queue limit is 1, queue limit check is not done for the 1st cycle alone, check for 2nd cycle is done during 1st cycle after sending 1st cycles data to child).
* If queue limit is reached for either of the child processes, then parent blocks on read (waits from acknowledgement from the child who queue limit is reached). Until, acknowledgement is received, the parent will not unblock.
* Once acknowledgement is received, parent checks the MSB bit of the integer sent as acknowledgement from child. If it is set, then error has occurred at child’s end. That instruction is ignored and parent moves to read next line in the instruction file.
* Once EOF is reached, parent quits the while loop and checks if any acknowledgement is to be received for sent messages, if yes, then waits in respective child’s queue to receive all acknowledgements.
* Once all acknowledgment are received (except for the “fac 0 0” and “add stopstop 0” instructions), parents waits for the child processes to exit. Once child processes exit, parent will exit the program.

**Adder Process:**

* Reads data from parent in a while loop.
* If received filename is “stopstop”, then adder exits without replying to parent.
* Else opens the file. If file is not found, then replies with an error message with MSB bit of reply data set to 1, other bits carry message count.
* Else reads the first line to get the number of integers to be added.
* Then reads all the integers from file in a loop and adds it to a sum variable.
* Once all integers are read and added to the sum, it exits the loop.
* Sum is a ‘double’ type variable, it is used to handle integer overflows. (‘int’ is 4 bytes and double is ‘8 bytes’).
* The output is displayed and adder sends the acknowledgement to the parent with the message count as data.
* Then waits in queue for next instruction.

**Factorial Process:**

* Reads data from parent in a while loop.
* If received integer values are zeroes, then adder exits without replying to parent.
* Else, calls fmod() function to calculate the factorial of number1 and (factorial(number1) mod number2). Then, prints the output.
* fact variable in facmod() is ‘double’ type, it is used to handle integer overflows. (‘int’ is 4 bytes and double is ‘8 bytes’).
* Factorial then sends the acknowledgement to the parent with the message count as data.
* Then waits in queue for next instruction.

Pipe 1

Pipe 4

Pipe 3

Pipe 2

Factorial

Adder

Parent

**Error Handling limitations:**

* Filename in add instruction cannot be more than 8 bytes.
* Integer values cannot be more than 2^30.
* No more factorial instructions in instruction.dat, once “fac 0 0” is sent.
* No more addition instructions in instruction.dat, once “add stopstop 0” is sent.
* Parent will not know what data is sent to child, it just checks if the operation is fac or add. It does not know if “stopstop” or “filename” is sent.
* Parent waits for acknowledgement only if queue limit is reached.
* Parent will block once EOF is reached, if child blocks without sending an acknowledgement for a previously sent instructions.
* Parent will not exit unless it receives all acknowledgements (except for “fac 0 0” and “add stopstop 0”). It will not exit if either of the child process is still alive.

**Data structures used:**

* An acknowledgment has the message count sent back to parent. It is simply a count of number of instructions received so far.
* An error acknowledgment has the MSB bit of the count value set to 1. Remaining bits will hold the count value.
* Parent will check the MSB, and act accordingly.
* facmod() function is used to calculate factorial and modulus.