

System Programming 2023 : Programming HW2 Report

Name : 張閔堯

ID : B11902084

General Implementation

Hierarchical Relationship

Each service is a process, but we only generate **Manager** in the terminal.

The hierarchical relationship is : if **A spawns B**, then A is B's parent process, that is, process A forks process B. Processes use pipe to communicate to each other, and each process has two pipes to communicate to its parent (for writing and reading, and Manager is an exception), and for each child, it use two pipes, too.

Each process has an array composed of the structure named **service** (name, write/read fds, pid), each element represents a child process. While doing pre-order traversal, just send message in the order of spawning.

The Philosophy for IPC

While passing command to its child, it has to block there and wait for its child to send message to it thru pipes, and it has to tell its parent that it has complete its mission, this philosophy ensure the order of commands. (if it's not manager)

The Delivery of Command

The manager receive command from **stdin**, and it use a structure to store the command, and pass it down to children. If a child receives the command, it would do one of the following :

1. Pass the command down to its children, have to block until receive message from children, if the child has successfully executes it, send a message to its parent to reach the early stop principle. (if it's not **Manager**)
2. Executes the command, send a message that describe success to its parent if it's not **Manager**.

Spawn : spawn A B

The **Manager** receives the message, pass it down to children. When a child receive the spawn message, it would check if A matches the service name.

If the service name matches, it would fork a new child, using two pipes to communicate with the child. In the end, send message to parent whether the spawn is failed or success, if manager receives failure, it would print **doesn't exist**.

Kill : kill K

Kill

A process receives the message from its parent (Manager : from `stdin`), it would see if the first child's name is kill K.

If yes, send a message `suicide` to kill K and `wait(NULL)`, and send a message that describes success to its parent if it's not **Manager**.

If not, send `kill` to kill K and After K receives the message, K would do the same.

And then if receive failure message from K, check the second child, if all child receive failure, send a failure message to its parent (**Manager**: print doesn't exist)

If it receives success message, print the number of killed children, and the services that are positioned in the behind of the killed one are re-indexed.

Suicide

If a process receives `suicide`, it would send the same message with pre-order traversal.

If a process receives this message and has no child, it would exit and tell its parent there is 1 process being killed.

And its parent would close the pipe to communicate with it, with the purpose of preventing extra use of fds. Besides, the parent would wait it in order to prevent zombie process.

After all its children are killed, it would count how many process are killed according to the message it receives and plus itself, and tell its parent the exact number.

Exchange : exchange A B

The main idea is : Do traverse until both services are found, print the messages. And then according to the name, find the corresponding pipe to the child, and then do exchange with FIFO.

Exchange

Manager firstly make FIFOs, and then do pre-order traverse to print messages, then send the exchange message to children, and receive messages from children to identify which is the correct pipe to communicate with the desired service.

In the end, **Manager** would wend `StartExchange` thru the pipes that leads to A and B, and after receiving the messages from the pipes, unlink the FIFOs and print the success message.

If the service name matches, then send success message to parent to let parent know which is the correct pipe to communicate with the desired children.

StartExchange

There are many trivial cases that describes the situations, but the final purpose is to find the correct child , after finding both, open the FIFOs and exchange their secrets.