

IPC

1. INTRODUCTION

In this report, I will focus on the 4 different way of IPC. I explain how the program works and what the difference is between those 4 methods.

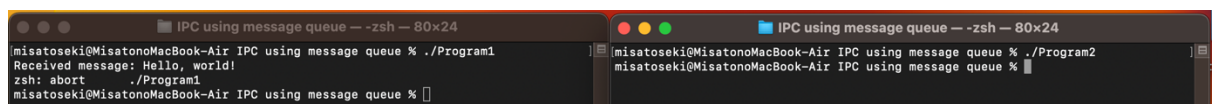
2. “IPC USING MESSAGE QUEUE”

This program executes inter-process communication (IPC) by using message queues. Message queue is a method of sending and receiving data between different processes. It deposits the data in a third area so that both sender and receiver can send and receive at their desired timing.

The first program creates a message queue and receives messages from the queue. First, a key is created, and a message queue is created using “msgget()” function. Then, it receives and displays messages from the queue using “msgrcv()” function. Finally, it deletes the message queue using “msgctl()” function.

The second program creates a message queue and sends messages to the queue. First, it creates a key, and a message queue is created using the msgget function. Then it sends a message to the queue using “msgsnd()” function.

The following picture shows how this program is executed.



```
misatoseki@MisatonoMacBook-Air IPC using message queue --zsh -- 80x24
misatoseki@MisatonoMacBook-Air IPC using message queue % ./Program1
Received message: Hello, world!
zsh: abort ./Program1
misatoseki@MisatonoMacBook-Air IPC using message queue %

misatoseki@MisatonoMacBook-Air IPC using message queue --zsh -- 80x24
misatoseki@MisatonoMacBook-Air IPC using message queue % ./Program2
misatoseki@MisatonoMacBook-Air IPC using message queue %
```

3. “IPC USING NAMED PIPE”

This program executes inter-process communication (IPC) by using named pipe. Named pipe is a method of inter-process communication (IPC) that is commonly used in UNIX-based operating systems.

The first program creates a named pipe and writes messages to it. It creates a pipe named /tmp/mypipe using “mkfifo()” function. Then it opens that pipe using “open()” function and writes the message “Hello, world!” using “write()” function.

The second program creates a named pipe and reads it. First, it creates a pipe named /tmp/mypipe using “mkfifo()” function. Then it opens that pipe using “open()” function and reads and displays the messages in the pipe using “read()” function.

The following picture shows how this program is executed.



```
misatoseki@MisatonoMacBook-Air IPC using named pipe % ./Program1
misatoseki@MisatonoMacBook-Air IPC using named pipe %

misatoseki@MisatonoMacBook-Air IPC using named pipe % ./Program2
Received message: Hello, w???
```

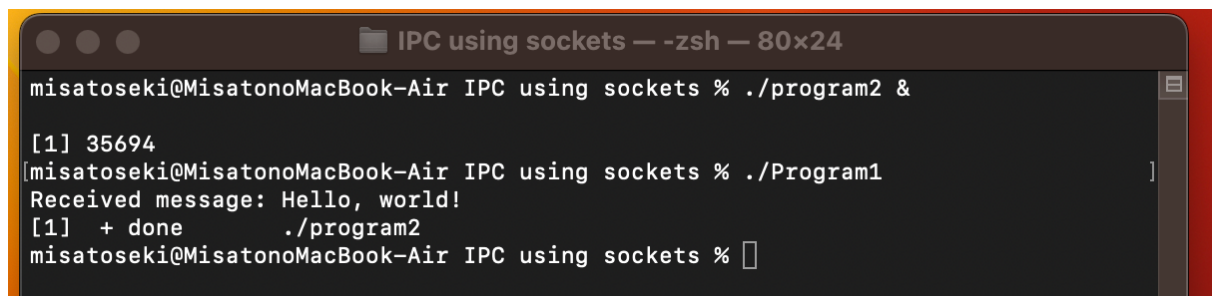
4. “IPC USING SOCKETS”

This program executes inter-process communication (IPC) by using sockets. Socket communication is a method that sets up a network such as IP addresses, port numbers and so on, sets up a server, and waits for connections from clients.

The first program creates a socket, connects to the specified address, and sends a specific message. First, the socket is created using “sockfd”. Next, it sets the server address including IP address and port number using “memset()” function. Finally, it connects to the socket using “connect()” function and sends messages using “sprintf()” and “write()” functions.

The second program creates a socket, listens for connection requests, and displays messages received. First, the socket is created using “sockfd”. Next, it sets the server address including IP address and port number using “memset()” function. Next, it listens for connection requests using “listen()” function. When a connection request is received, a message is read from the socket and displayed.

The following picture shows how this program is executed.



```
misatoseki@MisatonoMacBook-Air IPC using sockets % ./program2 &
[1] 35694
misatoseki@MisatonoMacBook-Air IPC using sockets % ./Program1
Received message: Hello, world!
[1] + done ./program2
misatoseki@MisatonoMacBook-Air IPC using sockets %
```

5. “IPC USING TCP SOCKET”

This program executes inter-process communication (IPC) by using TCP socket. TCP socket is a socket used to communicate between two processes to ensure reliable communication.

The first program deals with server-side socket programming. First, the socket is created using “sockfd”. Then it sets the server address using “memset()” function and binds the socket using “bind()” function. Next, the socket is listened for connection requests from clients using “listen()” function. When a connection request is received from the client, the socket is connected. Once the socket connection is established, text messages can be sent and received between the two processes by the code below “while(true)”.

The second program deals with client-side socket programming. First, a socket is created using “sockfd”. Then it obtains the IP address of the server and sends a connection request by the code below “server = gethostbyname(“localhost”);”. Once a connection is established to the server, text messages can be sent and received between the two processes by the code below “while(true)”.

The following picture shows how this program is executed.



The image shows two terminal windows side-by-side. The left window, titled 'IPC using TCP socket — Program1 — 80x24', displays the following text: 'misstoseki@MisstonoMacBook-Air IPC using TCP socket % ./Program1', 'Starting server...', 'Connected to client.', 'Client: Hello!', and 'Server: Hello!'. The right window, titled 'IPC using TCP socket — Program2 — 80x24', displays: 'misstoseki@MisstonoMacBook-Air IPC using TCP socket % ./Program2', 'Enter message: Hello!', 'Server replied: Hello!', and 'Enter message: ' followed by a cursor.

6. WHAT I LEARNED

I found that there are various methods for data communication between different processes. Message queue can pass data between multiple processes by sharing a key. Pipe does not allow data to be passed between multiple processes, but the procedure is simpler. Sockets can communicate between different machines because it uses a network.