Linux Programming
Python System Programming

컴퓨터공학과 송석일
sisong@ut.ac.kr

Python Socket API

서버

```
server_socket = socket.socket(socket.AF_INET,
                              socket.SOCK_STREAM)

server_socket.bind(('localhost', 8080))

server_socket.listen(5)

client_socket, addr = server_socket.accept()

data = client_socket.recv(1024)

client_socket.sendall(b'Hello, Client!')

client_socket.close()

server_socket.close()
```

클라이언트

```
client_socket = socket.socket(socket.AF_INET,
                              socket.SOCK_STREAM)

client_socket.connect(('localhost', 8080))

client_socket.sendall(b'Hello, Server!')

data = client_socket.recv(1024)

client_socket.close()
```

```python
import socket

def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 9999))
    server_socket.listen(5)
    print('Server is listening on port 9999')

    while True:
        client_socket, addr = server_socket.accept()
        print('Connected by', addr)
        client_socket.sendall(b'Hello, Client!')
        data = client_socket.recv(1024)
        print('Received from client:', data.decode())
        client_socket.close()

if __name__ == '__main__':
    start_server()
```

```python
import socket

def start_client():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect(('localhost', 9999))
    data = client_socket.recv(1024)
    print('Received from server:', data.decode())
    client_socket.sendall(b'Hello, Server!')
    client_socket.close()

if __name__ == '__main__':
    start_client()
```

```python
import socket
import os

def handle_client(client_socket):
    try:
        command = client_socket.recv(1024).decode()
        if not command:
            return

        pid = os.fork()

        if pid == 0:
            os.dup2(client_socket.fileno(), 1)
            os.dup2(client_socket.fileno(), 2)
            os.execlp(command.split()[0],
                      *command.split())
        else:
            os.waitpid(pid, 0)
    except Exception as e:
        client_socket.sendall(f"Error: {str(e)}".encode())
    finally:
        client_socket.close()

def start_server():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind(('localhost', 9999))
    server_socket.listen(5)
    print('Server is listening on port 8080')

    while True:
        client_socket, addr = server_socket.accept()
        print('Connected by', addr)
        handle_client(client_socket)

if __name__ == '__main__':
    start_server()
```

```python
import socket

def start_client():
    client_socket = socket.socket(socket.AF_INET,
                                  socket.SOCK_STREAM)
    client_socket.connect(('localhost', 9999))

    command = input("Enter command to execute: ")
    client_socket.sendall(command.encode())

    result = client_socket.recv(4096).decode()
    print("Command output:\n", result)

    client_socket.close()

if __name__ == '__main__':
    start_client()
```

문제 1

두개의 자식 프로세스를 생성하고 두 자식 프로세스가 파이프를 통해 메시지를 송수신하는 프로그램을 작성하라.
첫번째 생성된 자식 프로세스는 “Hello” 를 전송하고 두번째 생성된 자식 프로세스는 “Hello”를 수신하고 출력하도록 하라.
부모 프로세스는 두 자식 프로세스가 모두 종료될때 까지 대기했다가 자식 프로세스들이 종료되면 종료되었음을 출력하라.

문제 2

두개의 자식 프로세스를 생성하고 두 자식 프로세스가 파이프를 통해 메시지를 송수신하는 프로그램을 작성하라.
첫번째 생성된 자식 프로세스는 ls 명령을 실행하고 이때 생성된 결과를 두번째 자식 프로세스로 파이프를 통해서 전송한다.
두번째 생성된 자식 프로세스는 wc명령을 실행하고 wc 명령의 입력은 첫번째 자식 프로세스가 전송하는 ls 명령의 실행결과를 파이프를 통해서 입력받아 실행되도록 한다.

문제 3

클라이언트가 소켓을 통해서 서버에 wc|ls 를 전송한다.
서버는 두개의 자식 프로세스를 생성하고 첫번째 생성된 자식 프로세스는 ls 명령을 실행하고 이때 생성된 결과를 두번째 자식 프로세스로 파이프를 통해서 전송한다.
두번째 생성된 자식 프로세스는 wc명령을 실행하고 wc 명령의 입력은 첫번째 자식 프로세스가 전송하는 ls 명령의 실행결과를 파이프를 통해서 입력받아 실행되도록 한다.
실행 결과는 클라이언트에 보내서 결과를 보여준다.