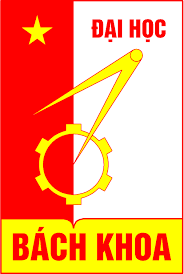
**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY**

\*\*\*\*\*



ITSS Embedded Linux

**Electric power supply control system**

*Instructor: MS. Banh Thi Quynh Mai*

|  |  |
| --- | --- |
| Student | Student ID |
| Hoang Tho Tung | 20184325 |
| Bui Thuc Nguyen Tien | 20180180 |
| Ha Quang Thieu | 20184309 |

*Hanoi, 8/2022*

# Abstract

Embedded Linux is a type of Linux operating system/kernel that is designed to be installed and used within embedded devices and appliances. In this Embedded Linux System course, we learn about linux embedded system concept, some important implementations and its applications. In this project, we build an Electric power supply control system. The implementation of this system helps us review the knowledge learned in this subject and some other related subjects. At the same time understand more about the role of the subject as well as its important applications, know how to deploy linux embedded systems.

Table of Contents

[Abstract 2](#_Toc79502253)

[A. Introduction 5](#_Toc79502254)

[I. Introduction about Electric power supply control system 5](#_Toc79502255)

[II. Division of work and contribution of each member 6](#_Toc79502256)

[B. Problem overview 9](#_Toc79502257)

[I. Outline 9](#_Toc79502258)

[II. Basic design specification of electric power supply control system 9](#_Toc79502259)

[III. Equipment specification 9](#_Toc79502260)

[IV. System specification 10](#_Toc79502261)

[C. Theoretical Background 11](#_Toc79502262)

[I. TCP programming in C 11](#_Toc79502263)

[II. Child process 12](#_Toc79502264)

[III. System V message queues 13](#_Toc79502265)

[D. Implementation 15](#_Toc79502266)

[I. Action Diagram 15](#_Toc79502267)

[I. Server 15](#_Toc79502268)

[II. Client 16](#_Toc79502269)

[E. Result 17](#_Toc79502270)

[F. Conclusion and Remark 19](#_Toc79502271)

[I. Conclusion 19](#_Toc79502272)

[II. Remark 19](#_Toc79502273)

List of Figures

[Figure 1 - System outline chart 5](#_Toc79502274)

[Figure 2 - Basic design specification of electric power supply control system 9](#_Toc79502275)

[Figure 3 - Command ipcs -l 14](#_Toc79502276)

[Figure 4 - System Diagram 15](#_Toc79502277)

[Figure 5 - The Server: Electrical System Management 17](#_Toc79502278)

[Figure 6 - The Client: Device Simulation 18](#_Toc79502279)

# A. Introduction

## I. Introduction about Electric power supply control system

We develop the electric power supply control system (It is described thereafter, "Electric power system"). The main function of the electric power system is as follows.

* The electric power system watches and controls the power supply.
* The electric power system supplies the electric power to the connected electric equipment (It is described thereafter as "Equipment").
* The electric power system records the electric power supply history.

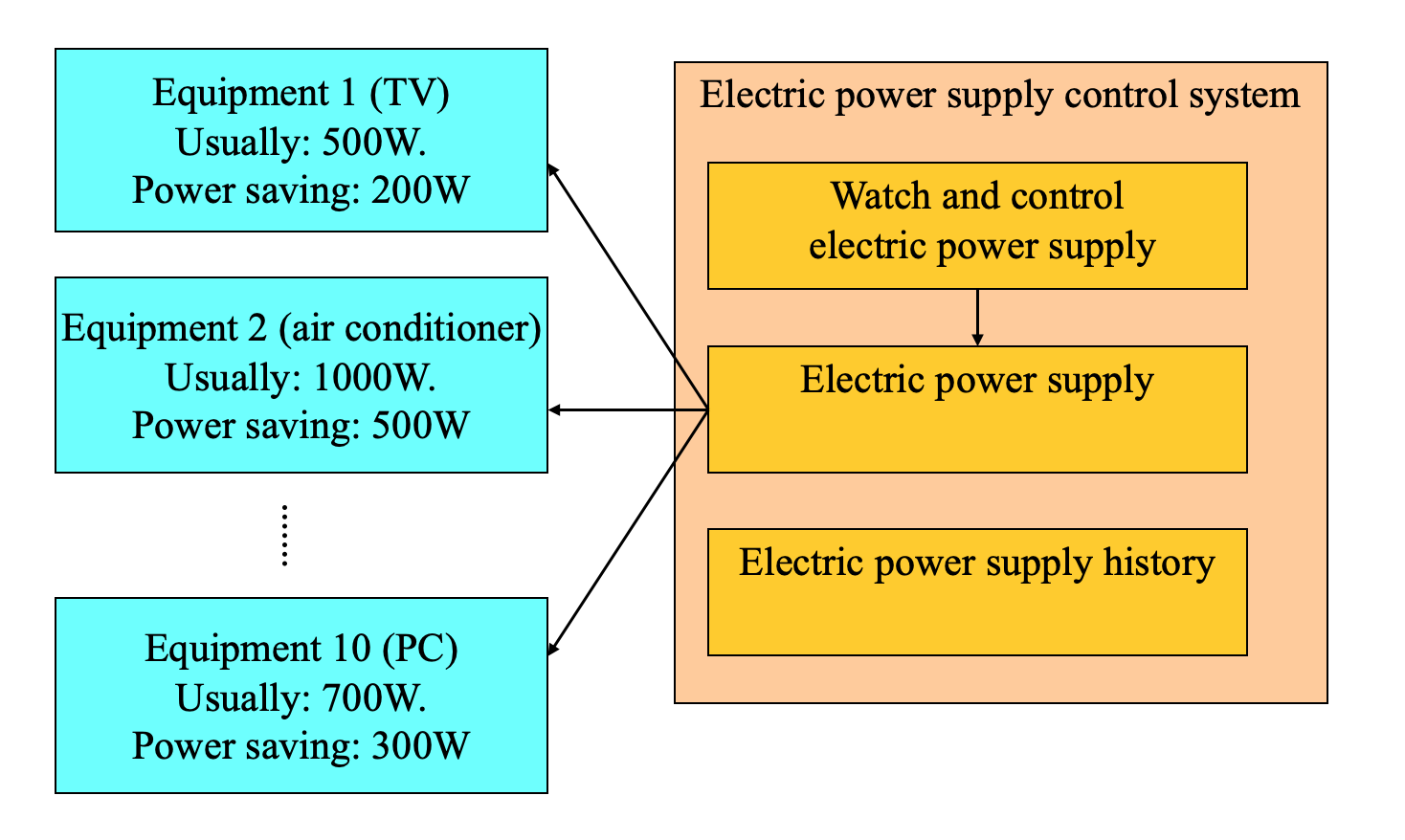


Figure 1 - System outline chart

When electric equipment such as TV, air conditioners, and personal computers connect it with the electric power system, the electric power system supplies the electric power corresponding to the use mode to the equipment. Then, it becomes possible to use equipment.

Detailed explanation of each mode:

1. The quantity of electric power used in the turning off mode is zero. As a result, the device is inoperable.
2. Ordinary mode refers to a mode that uses a standard amount of electric power. As a result, all of the equipment's functionalities are available.
3. The power saving mode is a mode that uses a smaller quantity of electricity than the regular mode. As a result, a portion of the equipment's functionality is restricted.

## II. Division of work and contribution of each member

|  |  |  |
| --- | --- | --- |
| Member | Work | Percentage |
| Ha Quang Thieu | - Design structs for the system: power system struct, device struct, message struct.  - Design the function powerSupply\_handle to write power supply information on the equipment. Moreover, powerSupply regularly notifies eleEquip the power supply and the status.  - Design the client.  - Write the report. | About 40% |
| Hoang Tho Tung | - System architecture design  - Design the function powSupplyInfoAccess\_handle to read and write power supply information according to the demand of the access of power supply information on another process.  - Design the function elePowerCtrl\_handle. When equipment is attached and disconnected, elePowerCtrl changes the power supply information on the equipment as well as the condition of the system's power supply based on the electric power supply situation.  - Write the report. | About 30% |
| Bui Thuc Nguyen Tien | - Design the function connectMng\_handle to create, and terminate child process (powerSupply) according to the connection and cutting from eleEquip.  - Design the function logWrite\_handle to write the log according to the demand to write the log from the other process.  - Write the report. | 30% |

The rest of this paper is designated as following: section B gives the problem overview; section C gives the theoretical background; section D gives details of the implementation; section E gives the result of the system and section F gives conclusion and remark.

# B. Problem overview

## I. Outline

Per the project requirement, the power supply control system has the following functionality:

* The electric power system watches and controls the power supply.
* The electric power system supplies the electric power to the connected electric equipment
* The electric power system records the electric power supply history.

## II. Basic design specification of electric power supply control system

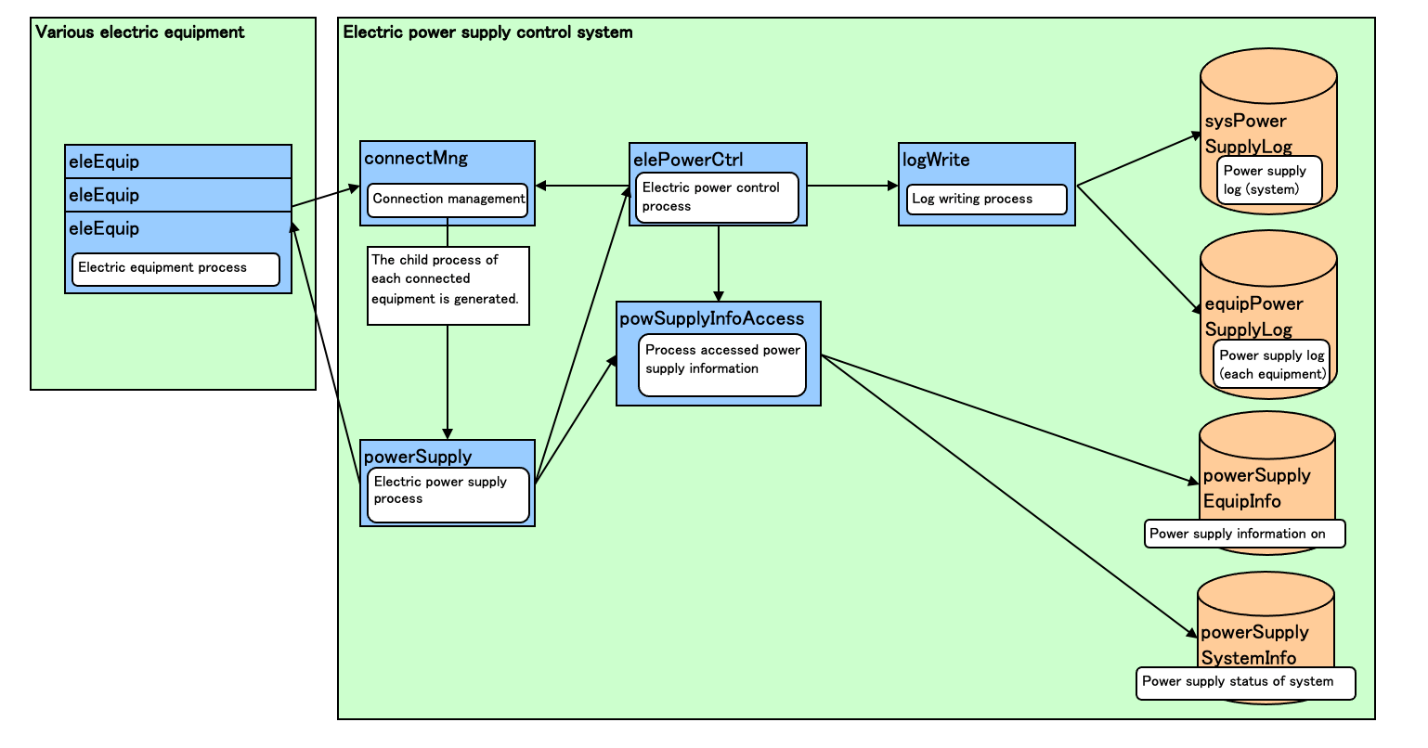


Figure 2 - Basic design specification of electric power supply control system

## III. Equipment specification

The equipment name and the use mode are set to each equipment.

There are three kinds of the following use mode of the equipment: turned-off mode, normal mode and electric power saving mode.

For each equipment, the power saving mode should consume less electricity than normal mode, and turned-off mode should consume no electricity.

The equipment can switch the use mode. When the equipment connects it with the electric power system, the use mode is assumed to be a turning off mode.

## IV. System specification

The equipment can be connected and cut for the electric power system. The maximum number that can be connected at the same time is assumed to be ten.

In the electric power system, the threshold of the amount of the electric power of the maximum supply is 5000W. The threshold in which warning is put out to the equipment is 4500W.

The electric power system notifies the warning to the connected equipment when the supplied electricity exceeds the warning threshold.

The electric power system observes the power supply to the equipment. When the electric power supply exceeds the maximum power supply (supply-over), the electric power system limits the power supply to the equipment to the quantity supplied of the power saving mode and prevent supply-over compulsorily.

If supply-over is resolved by the mode change and equipment cut, the electrical power system releases the limitation on the devices.

Though the electric power system limited the power supply, if over-supply cannot be prevented for a fixed time (ten seconds), the electric power system cut power to all devices.

Afterwards, when the over-supply is solved by cutting connected equipment, the electric power system restarts the supply of the electric power.

# C. Theoretical background

We program the system in C language, with the communication between the system and equipment is handled through TPC/IP and each task is handled by a different process.

## I. TCP programming in C

A TCP (transmission control protocol) is a connection-oriented communication. It is an intermediate layer of the application layer and internet protocol layer in the OSI model. TCP is designed to send the data packets over the network. It ensures that data is delivered to the correct destination.

TCP creates a connection between the source and destination node before transmitting the data and keeps the connection alive until the communication is active.

In TCP before sending the data it breaks the large data into smaller packets and cares the integrity of the data at the time of reassembling at the destination node. Major Internet applications such as the World Wide Web, email, remote administration, and file transfer rely on TCP.

TCP also offers the facility of retransmission, when a TCP client sends data to the server, it requires an acknowledgment in return. If an acknowledgment is not received, after a certain amount of time transmitted data will be loss and TCP automatically retransmits the data.

The communication over the network in TCP/IP model takes place in form of a client-server architecture: the client begins the communication and establishes a connection with a server.

Steps to create a client using TCP/IP:

* Create a socket using the socket() function in c.
* Initialize the socket address structure as per the server and connect the socket to the address of the server using the connect()
* Receive and send the data using the recv() and send() functions.
* Close the connection by calling the close() function.

Steps to create a server using TCP/IP:

* Create a socket using the socket() function in c.
* Initialize the socket address structure and bind the socket to an address using the bind() function.
* Listen for connections with the listen() function.
* Accept a connection with the accept() function system call. This call typically blocks until a client connects to the server.
* Receive and send data by using the recv() and send() function in c.
* Close the connection by using the close() function.

## II. Child process

A child process is the creation of a parent process, which can be defined as the main process that creates child or subprocesses to perform certain operations. Each process can have many child processes but only one parent. A child process inherits most of its parent's attributes.

A parent process can create multiple child processes. If a process does not have a parent, it is assumed to be created directly by the kernel.

In systems such as Unix and Linux, the first process, "init", is created by the kernel at boot time and never terminated as long as the system is running. Other parentless processes may be launched to perform different daemon tasks.

In some situations, a child process is orphaned when its parent dies. The orphaned child process is then shortly adopted by the init process.

When a child process is created, it is associated with a unique process ID number. The lifetime of a process ends when a termination signal is reported to the parent process, resulting in the release of the process ID and resources.

Fork system call is used for creating a new process, which is called child process, which runs concurrently with the process that makes the fork() call (parent process). After a new child process is created, both processes will execute the next instruction following the fork() system call. A child process uses the same pc(program counter), same CPU registers, same open files which use in the parent process.

It takes no parameters and returns an integer value. Below are different values returned by fork():

* Negative Value: creation of a child process was unsuccessful.
* Zero: Returned to the newly created child process.
* Positive value: Returned to parent or caller. The value contains process ID of newly created child process.

## III. System V message queues

Message queues are one of the inter-process communication mechanisms available under Linux. It is a method by which process (or program instances) can exchange or pass data using an interface to a system-managed queue of messages. Messages can vary in length and be assigned different types or usages. A message queue can be created by one process and used by multiple processes that read and/or write messages to the queue.

Message queues, shared memory and semaphores are normally listed as the three inter-process communication mechanisms under Linux. Semaphores, though, are really for process synchronization. In practice, shared memory, aided by semaphores, makes an inter-process communication mechanism. Message queues is the other inter-process communication mechanism.

There are two varieties of message queues, System V message queues and POSIX message queues. Both provide almost the same functionality but system calls for the two are different. System V message queues have been around for a long time, since the UNIX systems of 1980s and are a mandatory requirement of Unix-certified systems. POSIX message queues (and the complete POSIX IPC calls) were introduced in 1993 and are still an optional requirement of Unix-certified systems.

There are three system wide limits regarding the message queues. These are, MSGMNI, maximum number of queues in the system, MSGMAX, maximum size of a message in bytes and MSGMNB, which is the maximum size of a message queue.

The command ipcs -l show relevant information about message queues limitation, as well as other inter-process communication method.

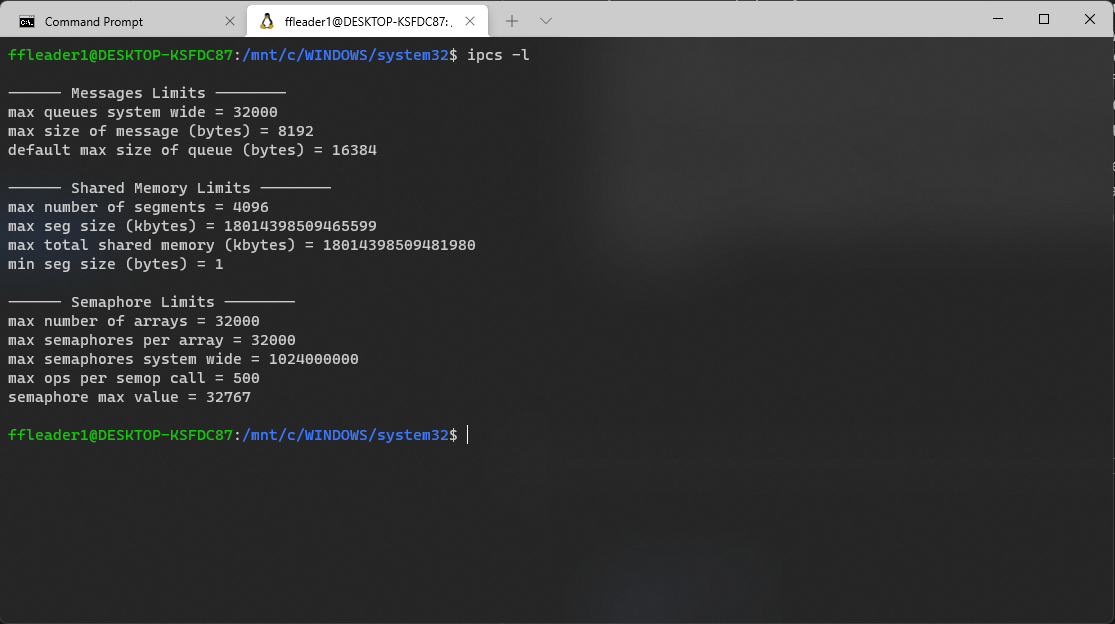


Figure 3 - Command ipcs -l

# D. Implementation

## I. Action Diagram

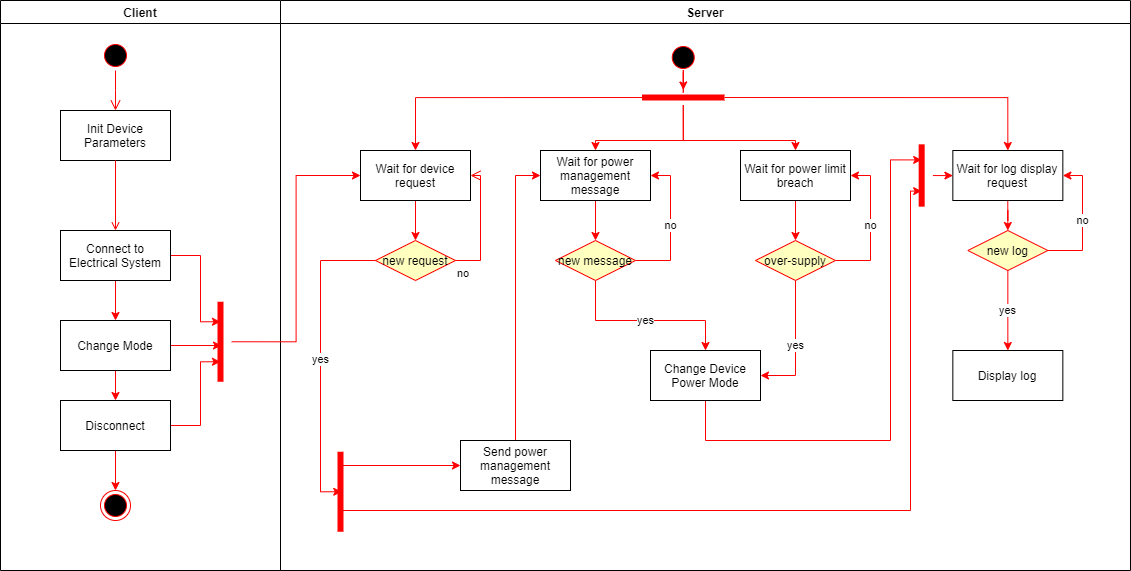


Figure 4 - System Diagram

The action diagram for the system can be expressed as above.

When a device is initialized, change mode or disconnected, a message will be sent to the listening server through TCP/IP.

In the server, for separated process will be running simultaneously for listening to device request, make power adjustment from device request, prevent over-supply and display log.

## I. Server

The sever receive connection signal from incoming devices, manage power and write relevant log.

There are five prominent functions in our source codes whose functionality can be described as follow:

* powerSupply\_handle(): From the message received by the client, this function would send a corresponding command to active electrical device process.
* elePowerCtrl\_handle(): Monitor existing system condition and send necessary message to invert supply-over situations.
* powSupplyInfoAccess\_handle(): Adjust the electrical device based on received message passing between process. These message are send from the two above functions, which are power-management command message, initialized by either the user or the system, respectively. The message header can ne ‘n’, ‘m’ or ‘d’, which is for creating new device, changing mode of a connecting device and disconnect a device, respectively.
* connectMng\_handle(): Handle communication with simulated device from the client(s).
* logWrite\_handle(): Write log file

## II. Client

The client represents a simulated device connecting to the system.

When initialized, the programs require 3 inputs: device name, device power in normal mode, and device power in limited mode.

If user inputs power numbers whose values in limited mode is more than normal mode, the program would as the user to input again.

When the connection is established, the user will have the option to change the device mode (turned-off, normal or limited) or disconnect it.

# E. Result

We have simulated a power supply control system with features as described by the problem.

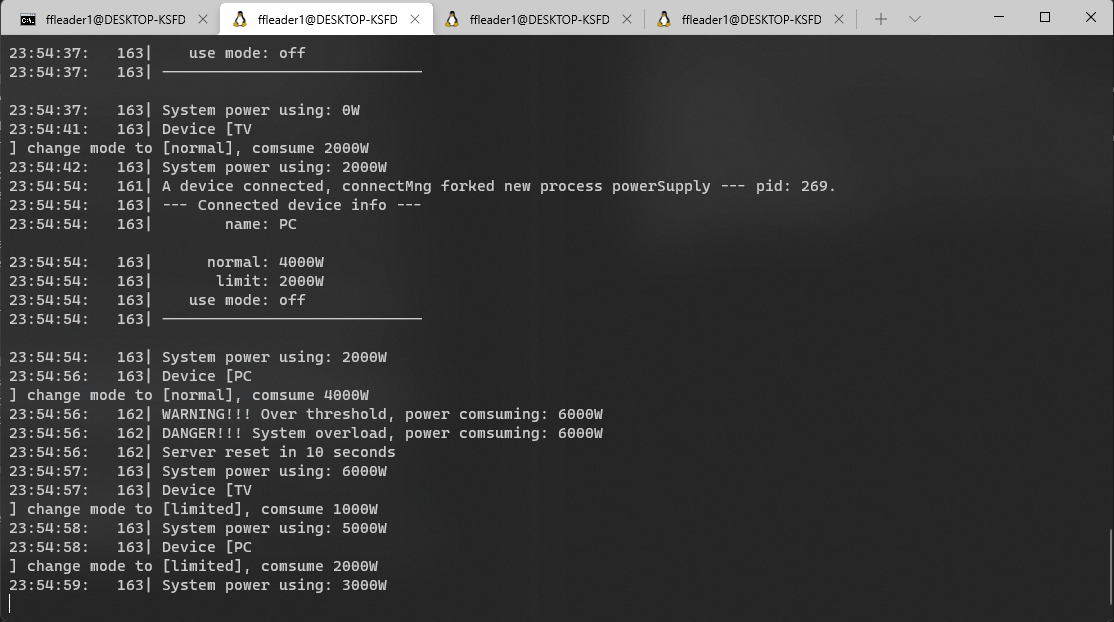


Figure 5 - The Server: Electrical System Management

The server handles incoming connection and perform necessary task to handle over-supply.

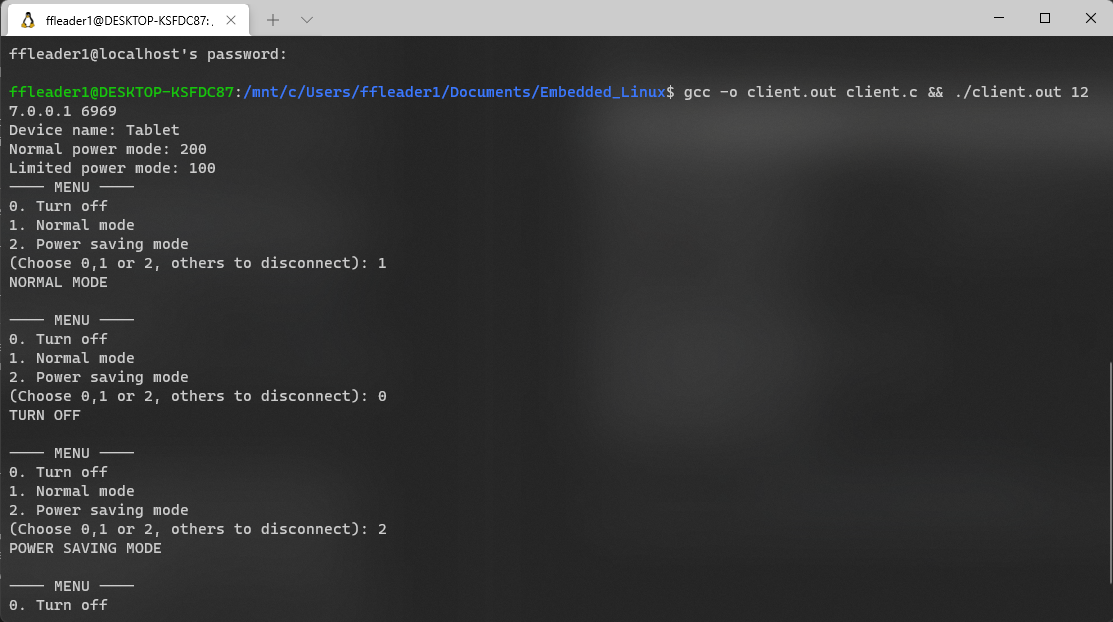


Figure 6 - The Client: Device Simulation

The clients connect to the server and change between various power mode.

# F. Conclusion and Remark

## I. Conclusion

We have constructed an electrical system in C that would adjust power delivery to its simulated device, either by user manual input or system automatic adjustment to prevent over-supply. The system contains two separated programs communicated through TCP/IP. The client represents a connecting device and change its power mode based on user input. The server manages the electrical system with various functions run in separated processes to handle user request or prevent over-supply.

## II. Remark

Through this project, we have learned about TCP/IP protocol and able to program certain pieces of software that communicate through the mentioned protocol. Also, we have explored multiprocessing in C to make our system more efficient. This knowledge can be implemented in the future to make more practical and accessible programs.