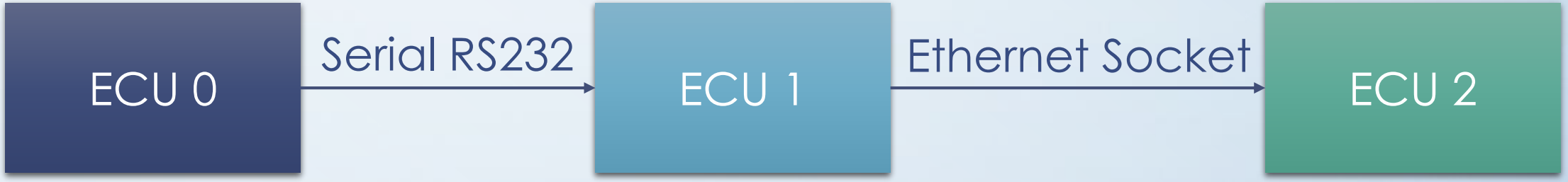
Embedded Linux Course

Socket Project

Presented by:
Ahmed Nader

Top-level System Topology



ECU-0

- Dummy ECU in our system.
- It is used to send packets to ECU-1 via RS232 serial interface.

ECU-1

- It is used to receive packets from ECU-0 via RS232 serial interface.
- Validate the received data from ECU-0.
- Add the received time to the packet.
- Send the packet to ECU-2 via Ethernet Socket communication (server).
- If ECU-1 terminates,
  - It closes socket connections.
  - It closes serial connections.

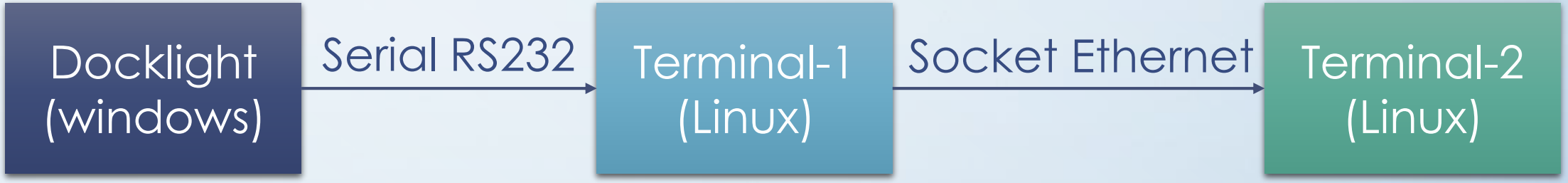
ECU-2

- It is used to receive packets from ECU-1 via Ethernet Socket communication (client).
- Add the received time to the packet.
- Dump the received packet into file in ECU-2's system files.
- If ECU-2 terminates,
  - It closes socket connection.
  - It creates a child process to show the dumped file to the user.

Packet life cycle

Implementation Phase

Mapping system topology



“canPacket” Union

```
/********************** Definitions *********************************/
#define TIME_SIZE       24
#define DATA_MAX_SIZE   64

/********************** Decelerations *******************************/
#pragma pack(push,1)
struct packet_t {
    unsigned short ID;
    unsigned char DLC;
    char data[DATA_MAX_SIZE + TIME_SIZE];

} packet;
#pragma pack(pop)

typedef union {
    struct packet_t packet;
    char packetInBytes[sizeof(packet)];
} canPacket;

/****************************** externs *****************************/
```

ECU-1

```
/********************** Definitions *********************************/
#define TIME_SIZE       24
#define DATA_MAX_SIZE   64

#define FILE_PATH "/home/embedded_system_ks/workspace/client_socket/dataLog.txt"
/********************** Decelerations *******************************/

#pragma pack(push,1)
struct packet_struct {
    unsigned short ID;
    unsigned char DLC;
    char data[DATA_MAX_SIZE + 2*TIME_SIZE];

}packet;
#pragma pack(pop)

typedef struct packet_struct packet_t;

typedef union {
    packet_t packet;
    char packetInBytes[sizeof(packet)];
} canPacket;

extern pthread_mutex_t recvMutex;
extern int file_fd;
extern canPacket receivedPacket;
```

ECU-2

ECU-1 Threads

```c
int main(int argc, char * argv[]) {

    pthread_t serial_thread;
    pthread_t dataValidation_thread;
    pthread_t socketServer_thread;

    serial_init(argv[1]);

    pthread_mutex_lock(&serialData_mutex);
    pthread_mutex_lock(&dataValidation_mutex);

    signal(SIGINT, int_signal_handler);

    if ((pthread_create(&serial_thread, NULL, serial_read, NULL)) < 0) {
        perror("serial_thread_create");
        exit(0);
    }
    if ((pthread_create(&dataValidation_thread, NULL, data_validation, NULL))
            < 0) {
        perror("dataValidation_thread_create");
        exit(0);
    }
    if ((pthread_create(&socketServer_thread, NULL, socket_server, NULL)) < 0) {
        perror("server_thread_create");
        exit(0);
    }
    pthread_join(serial_thread, NULL);
    pthread_join(dataValidation_thread, NULL);
    pthread_join(socketServer_thread, NULL);

}
```

```c
void * serial_read(void * arg) {

    while (1) {
        printf(".....Inside serial_read thread.....\n");

        bytes_read = read(serial_fd, (char *) recievedPacket.packetInBytes,
                sizeof(recievedPacket.packetInBytes));
        //clear buffer
        memset(recievedPacket.packetInBytes + bytes_read, 0x00,
                sizeof(recievedPacket.packetInBytes) - bytes_read);
        //Echo received bytes again
        write(serial_fd, recievedPacket.packetInBytes, strlen(recievedPacket.packetInBytes));

        printf("received packet is = %s\nand number of bytes are %d\n",
                recievedPacket.packetInBytes, bytes_read);

        pthread_mutex_unlock(&serialData_mutex);
    }
}
```

```c
void * data_validation(void * arg) {

    time_t Epoch_date = 0;

    while (1) {
        pthread_mutex_lock(&serialData_mutex);

        printf(".....Inside data_validation thread.....\n");

        if (((recievedPacket.packetInBytes[2] - '0')
                == strlen(recievedPacket.packetInBytes) - 3)
                && ((recievedPacket.packetInBytes[2] - '0') == bytes_read - 3)) {

            printf("received data is valid\n");

            Epoch_date = time(NULL);

            memcpy(recievedPacket.packet.data + strlen(recievedPacket.packet.data),
                    ctime(&Epoch_date), TIME_SIZE);

            pthread_mutex_unlock(&dataValidation_mutex);

        } else {

            printf("received data is not valid\nPlease try again\n");
        }

    }
```

```c
void * socket_server(void *arg) {

    ethernetSocketServer_init();

    while (1) {

        pthread_mutex_lock(&dataValidation_mutex);

        printf("packet to be sent = %s\n", recievedPacket.packetInBytes);
        write(new_socket, recievedPacket.packetInBytes, strlen(recievedPacket.packetInBytes));

        printf("\n\n");

    }

}
```

ECU-2 Threads

```c
int main(int argc, char * argv[]) {

    pthread_t clientSocket_thread;
    pthread_t dumpFile_thread;

    //acquire mutex
    pthread_mutex_lock(&recvMutex);
    //Create threads
    signal(SIGINT, int_Signal_handler);

    if (pthread_create(&clientSocket_thread, NULL, Thread_SocketClient, NULL)
            < 0) {
        perror("clientSocket_thread");
        exit(0);
    }
    if (pthread_create(&dumpFile_thread, NULL, Thread_WriteFile, NULL) < 0) {
        perror("dumpFile_thread");
        exit(0);
    }

//join threads
    pthread_join(clientSocket_thread, NULL);
    pthread_join(dumpFile_thread, NULL);
}
```

```c
void * Thread_WriteFile(void *arg) {

    time_t Epoch_date = 0;

    file_fd = open(FILE_PATH, O_RDWR | O_CREAT | O_TRUNC, 0744);

    if (file_fd == -1) {
        perror("File Open");
        exit(0);
    }
    while (1) {

        pthread_mutex_lock(&recvMutex);

        printf("File Opened waiting data..... \n");

        Epoch_date = time(NULL);

        memcpy(receivedPacket.packet.data + strlen(receivedPacket.packet.data),
               ctime(&Epoch_date), 24);

        printf("writing : %s\n\n",receivedPacket.packetInBytes);


         write_in_file();
    }
}
```

```c
void * Thread_SocketClient(void * arg) {

    ethernetSocketClient_init();
    while (1) {
        while (!received_bytes) {
            received_bytes = read(client_fd, receivedPacket.packetInBytes,
                    sizeof(receivedPacket.packetInBytes));
        }
        memset(receivedPacket.packetInBytes + received_bytes, 0,
                sizeof(receivedPacket.packetInBytes) - received_bytes);

        printf("received packet: %s\nand number of bytes are %d\n",
                receivedPacket.packetInBytes, received_bytes);

        received_bytes = 0;

        //release mutex
        pthread_mutex_unlock(&recvMutex);
    }
}
```

Data log file example

```
ID = ab
DLC = 3
data = 012
TX time = Sat Dec 28 00:54:33 2019
RX time = Sat Dec 28 00:54:33 2019
=======================================
ID = av
DLC = 5
data = 01234
TX time = Sat Dec 28 00:54:35 2019
RX time = Sat Dec 28 00:54:35 2019
=======================================
ID = ab
DLC = 3
data = 012
TX time = Sat Dec 28 00:54:37 2019
RX time = Sat Dec 28 00:54:37 2019
=======================================
ID = av
DLC = 5
data = 01234
TX time = Sat Dec 28 00:54:39 2019
RX time = Sat Dec 28 00:54:39 2019
=======================================
```

thank you!