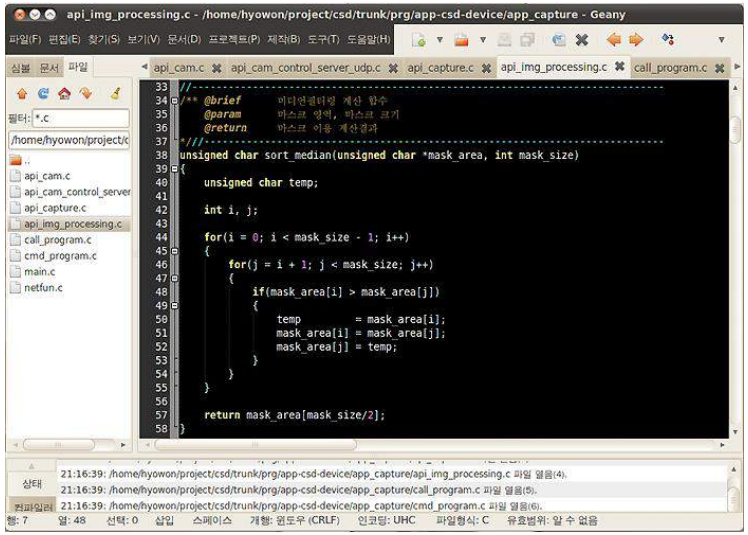
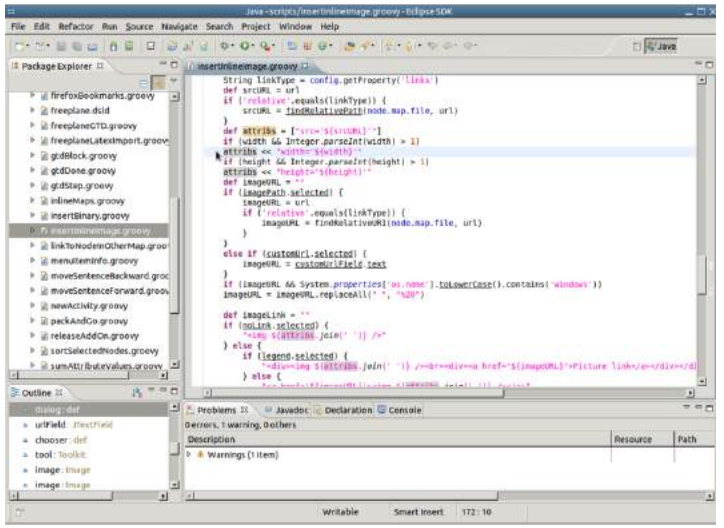
PPP Implementation & Serial Communication

Development environment

- Edit tool

1) geany - install command : apt-get install geany
2) vi (linux default edit) – it is difficult to use at first
3) eclips – install command : apt-get install eclips
4) share folder (in window, uses a visual-studio
   -> take it into Ubuntu using share folder
   -> compile it in Ubuntu (uses a gcc compiler)





Serial Communication

- Open serial device

```
#include <stdio.h>          #include <stdlib.h>
#include <fcntl.h>          #include <string.h>
#include <termios.h>

#define BAUDRATE B38400
#define SERIALDEVICE /dev/ttyS0

int main(void) {
    int fd;
    struct termios  tio;

    fd = open(SERIALDEVICE, O_RDWR | O_NOCTTY | O_NONBLOCK);
    if (fd < 0) { perror(SERIALDEVICE); exit(-1); }
    ...
}
```

Serial Communication

- Device control and file description
  - Given a pathname for a file(or a device), open() returns a file descriptor
  - The file descriptor is used in device control functions (system calls such as read(), write())
- The termios structure
  - data structure containing terminal configuration
  - c_cflag/c_iflag :control mode/input mode flags

Serial Communication

- Serial device settings

```
bzero(&tio, sizeof(tio));
tio.c_cflag = BAUDRATE | CRTSCTS | CS8 | CLOCAL | CREAD;
tio.c_iflag = IGNPAR | ICRNL;

tcflush (fd, TCIFLUSH);
tcsetattr (fd, TCSANOW, &tio);
```

  - tcflush(): discards data in the fd
    - TCIFLUSH : flushes data received but not read
  - tcsetattr(): sets the parameters associated with the terminal

Serial Communication

- Read/write serial device

```
int res, snd, buf_size;
unsigned char buf[255];

res = read(fd, buf, 255);
snd = write(fd, buf, buf_size);
```

  - read(int fd, void *buf, size_t count)
    - reads up to count bytes from a fd into the buf, returns the number of bytes successfully read
  - write(int fd, const void *buf, size_t count)

PPP Implementation

- AHDLC frame Receiving procedure
  - ① Some byte streams are arrived
  - ② Assemble sequential bytes into a AHDLC PPP frame (between flag/delimiter 7Es)
  - ③ Check the CRC using CRC-16 mechanism
  - ④ Remove escape codes
  - ⑤ Check protocol field
    - LCP : LCP negotiation process
    - IPCP : IPCP negotiation process

Example of AHDLC frame receiving procedure

① RCVD: 7e ff 7d 23 c0 21 7d 21 7d 21 7d 20 7d 2e
RCVD: 7d 22 7d 26 7d 20 7d 20 7d 20 7d 20 7d 27
RCVD: 7d 22 7d 28 7d 22 70 34 7e ff 7d 23 c0 21

② 7e ff 7d 23 c0 21 7d 21 7d 21 7d 20 7d 2e 7d 22 7d 26 7d 20 7d 20 7d 20 7d 20 7d 27 7d 22 7d 28 7d 22 70 34 7e

③ CRC check : 70 34 is good crc

④ 7e ff 03 c0 21 01 01 00 0e 02 06 00 00 00 00 07 02 08 02 70 34 7e

⑤ Protocol field : c0 21 -> LCP procedure

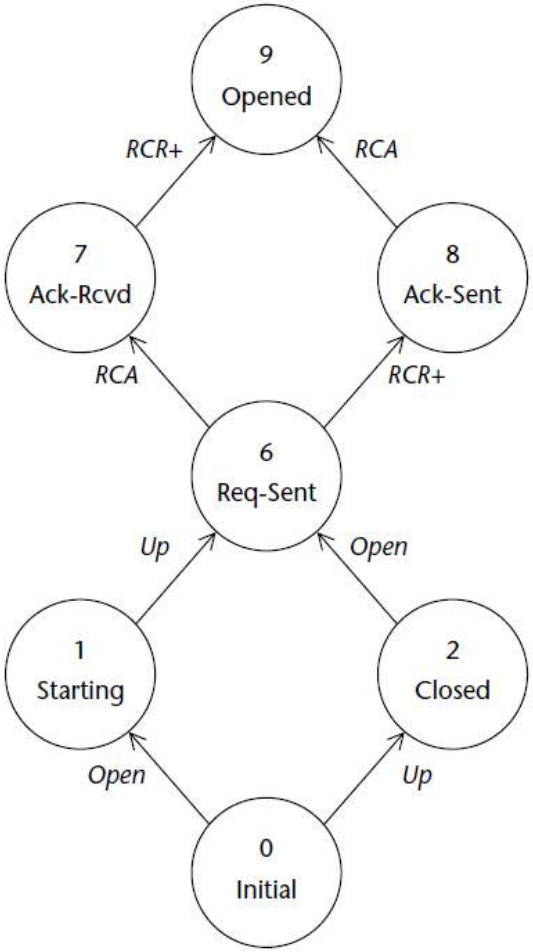
PPP Implementation

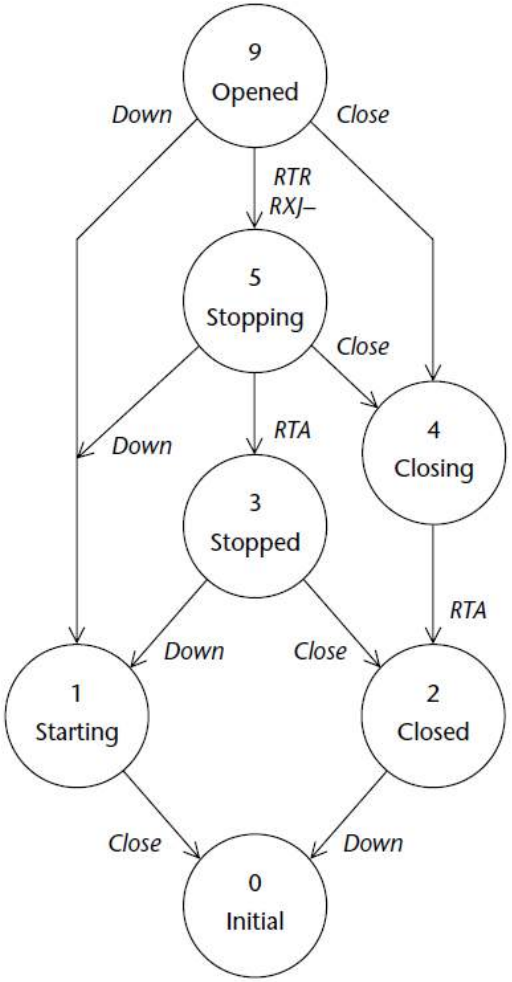
- AHDLC frame sending procedure
  - Get original data from upper protocols (LCP, IPCP)
  - Convert original data to escape code if ACCM is on
  - Append a CRC code
    - Convert to escape code if ACCM is on
  - Add flag/delimiter 7E to the frame

PPP Implementation

- LCP implementation
  - Must implement a PPP state machine
    - connection/termination transition must be covered
  - LCP negotiation procedure follows transition of the PPP state machine
  - LCP should be able to handle ACCM, pcomp, accomp options

PPP state machine

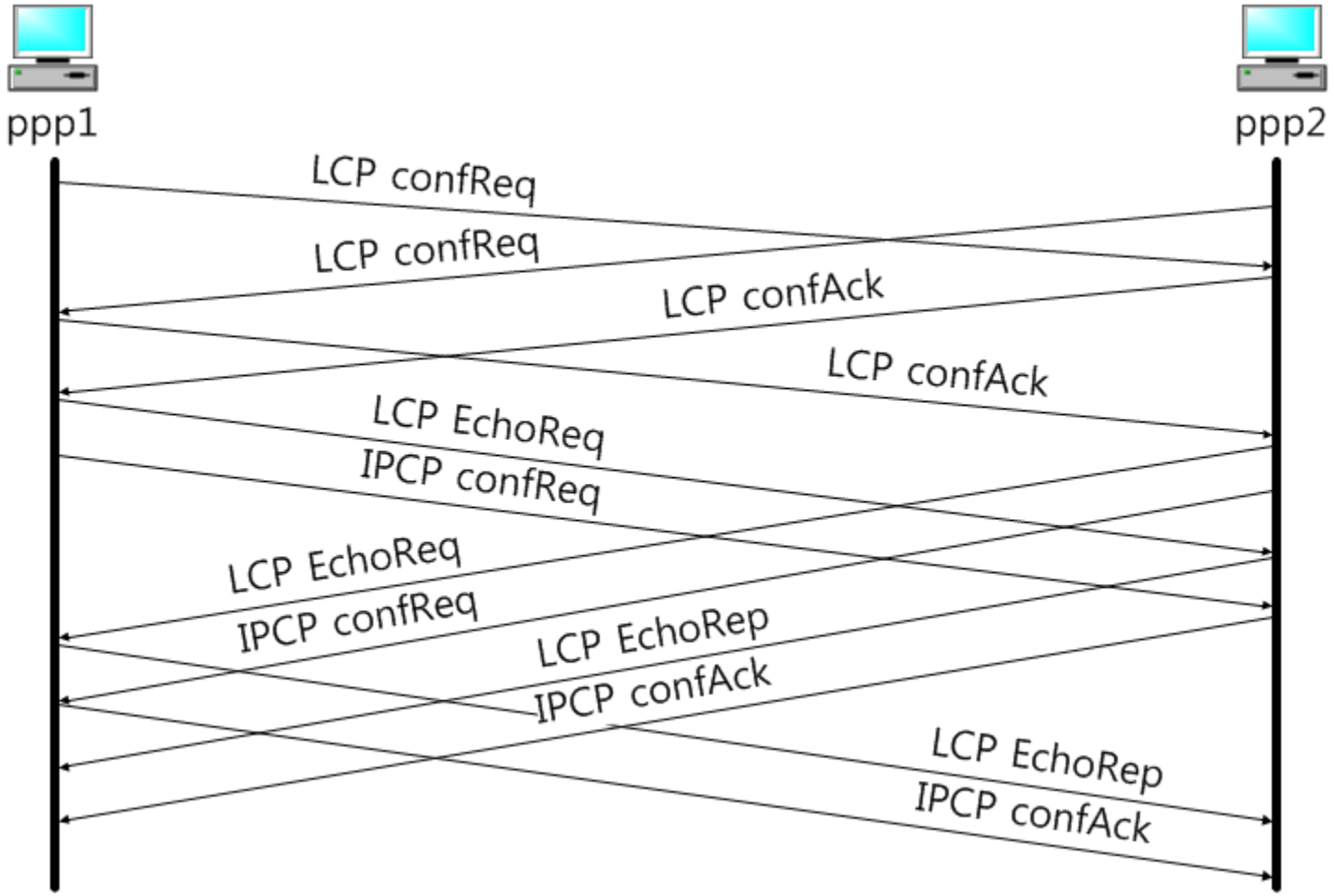




PPP state machine

	State							State			
	0	1	2	3	4	5		6	7	8	9
Events	Initial	Starting	Closed	Stopped	Closing	Stopping	Events	Req-Sent	Ack-Rcvd	Ack-Sent	Opened
Up	2	irc,scr/6	-	-	-	-	Up	-	-	-	-
Down	-	-	0	tls/1	0	1	Down	1	1	1	tld/1
Open	tls/1	1	irc,scr/6	3r	5r	5r	Open	6	7	8	9r
Close	0	tlf/0	2	2	4	4	Close	irc,str/4	irc,str/4	irc,str/4	tld,irc,str/4
TO+	-	-	-	-	str/4	str/5	TO+	scr/6	scr/6	scr/8	-
TO-	-	-	-	-	tlf/2	tlf/3	TO-	tlf/3p	tlf/3p	tlf/3p	-
RCR+	-	-	sta/2	irc,scr,sca/8	4	5	RCR+	sca/8	sca,tlu/9	sca/8	tld,scr,sca/8
RCR-	-	-	sta/2	irc,scr,scn/6	4	5	RCR-	scn/6	scn/7	scn/6	tld,scr,scn/6
RCA	-	-	sta/2	sta/3	4	5	RCA	irc/7	scr/6x	irc,tlu/9	tld,scr/6x
RCN	-	-	sta/2	sta/3	4	5	RCN	irc,scr/6	scr/6x	irc,scr/8	tld,scr/6x
RTR	-	-	sta/2	sta/3	sta/4	sta/5	RTR	sta/6	sta/6	sta/6	tld,zrc,sta/5
RTA	-	-	2	3	tlf/2	tlf/3	RTA	6	6	8	tld,scr/6
RUC	-	-	scj/2	scj/3	scj/4	scj/5	RUC	scj/6	scj/7	scj/8	scj/9
RXJ+	-	-	2	3	4	5	RXJ+	6	6	8	9
RXJ-	-	-	tlf/2	tlf/3	tlf/2	tlf/3	RXJ-	tlf/3	tlf/3	tlf/3	tld,irc,str/5
RXR	-	-	2	3	4	5	RXR	6	7	8	ser/9

Connection Establishment



Homework #2

- Implement PPP
  - Implement a basic PPP protocol included LCP, IPCP and connect to the pppd program to complete PPP connection establishment and termination.
  - Implementation components
    - AHDLC PPP framing with CRC-16 and character escaping process
    - LCP negotiations with PPP state machine (connection establishment/termination)
    - IPCP negotiations
    - Print hex codes of send/recv frame and analyze messages like debug of pppd

Homework #2

- Test & Result
  - Connect your developed program to a pppd of virtual machine
  - The program should be able to negotiate accomp, pcomp options.
- submission
  - Result screenshot
    - Execution results : both of pppd and your program results
    - Show hex codes, message exchanges, analysis of messages
  - Report
    - Explain your program : what components is implemented, especial points(functions, algorithms, codes, and so on)

References

- RFC 1661, RFC 1662, RFC 1332
- James Carlson, “PPP Design, Implementation, and Debugging 2nd edition,” Addison-Wesley, July 2000.
- http://www.tldp.org/HOWTO/Serial-Programming-HOWTO/