#include <errno.h>

#include <fcntl.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <termios.h>

#include <unistd.h>

#include <ncurses.h>

int set\_interface\_attribs(int fd, int speed)

{

struct termios tty;

if (tcgetattr(fd, &tty) < 0) {

printf("Error from tcgetattr: %s\n", strerror(errno));

return -1;

}

cfsetospeed(&tty, (speed\_t)speed);

cfsetispeed(&tty, (speed\_t)speed);

tty.c\_cflag |= (CLOCAL | CREAD); /\* ignore modem controls \*/

tty.c\_cflag &= ~CSIZE;

tty.c\_cflag |= CS8; /\* 8-bit characters \*/

tty.c\_cflag &= ~PARENB; /\* no parity bit \*/

tty.c\_cflag &= ~CSTOPB; /\* only need 1 stop bit \*/

tty.c\_cflag &= ~CRTSCTS; /\* no hardware flowcontrol \*/

/\* setup for non-canonical mode \*/

tty.c\_iflag &= ~(IGNBRK | BRKINT | PARMRK | ISTRIP | INLCR | IGNCR | ICRNL | IXON);

tty.c\_lflag &= ~(ECHO | ECHONL | ICANON | ISIG | IEXTEN);

tty.c\_oflag &= ~OPOST;

/\* fetch bytes as they become available \*/

tty.c\_cc[VMIN] = 1;

tty.c\_cc[VTIME] = 1;

if (tcsetattr(fd, TCSANOW, &tty) != 0) {

printf("Error from tcsetattr: %s\n", strerror(errno));

return -1;

}

return 0;

}

void set\_mincount(int fd, int mcount)

{

struct termios tty;

if (tcgetattr(fd, &tty) < 0) {

printf("Error tcgetattr: %s\n", strerror(errno));

return;

}

tty.c\_cc[VMIN] = mcount ? 1 : 0;

tty.c\_cc[VTIME] = 5; /\* half second timer \*/

if (tcsetattr(fd, TCSANOW, &tty) < 0)

printf("Error tcsetattr: %s\n", strerror(errno));

}

int main()

{

char \*portname = "/dev/ttyO4";

int fd;

int wlen;

unsigned char buf[80];

float sum=0.000, total=0.000;

//unsigned char assign[80];

//static const char barcode[10] = "180100005";

//static const char hdmi[6] = "39907";

int rdlen;

int ii=0;

printf("Welcome to minicom terminal\n");

fd = open(portname, O\_RDWR | O\_NOCTTY | O\_SYNC);

if (fd < 0) {

printf("Error opening %s: %s\n", portname, strerror(errno));

return -1;

}

/\*baudrate 115200, 8 bits, no parity, 1 stop bit \*/

set\_interface\_attribs(fd, B9600);

//set\_mincount(fd, 0); /\* set to pure timed read \*/

/\* simple output \*/

wlen = write(fd, "Hello!\n", 7);

if (wlen != 7) {

printf("Error from write: %d, %d\n", wlen, errno);

}

// tcdrain(fd); /\* delay for output \*/

/\* simple noncanonical input \*/

do {

// unsigned char buf[80] ;

int rdlen;

rdlen = read(fd, buf, sizeof(buf) - 1);

//printf("rdlen = %d",rdlen);

if (rdlen > 0)

{

// int i=2;

// buf[rdlen] = 0;

// assign[80]=buf[80];

//printf("value =

if(rdlen == 14)

{

printf("JAPANESE CHERRY BLOSSOM\n");

printf("fine fragrance mist - Bath & Body Works CANADA\n");

printf("$18.50");

sum = 18.50; }

else if(rdlen == 17)

{

printf("KOZMIC COLOURS\n");

printf("Nail Polish Set\n");

printf("$1.50");

sum = 1.50;

}

else if(rdlen == 32)

{

printf("SUBWAY Gift-Card\n");

printf("$25");

sum = 25.000;

}

else if(rdlen == 10)

{

printf("ANKLE SOCKS SOCQUTTES\n");

printf("ARDENE 4 PAIRS OF SOCKS 95% Polyster\n");

printf("$7.50");

sum = 7.50;

}

else if(rdlen ==11)

{

printf("CINTHOL DEO SPRAY\n");

printf("godrej product:no Alchohole ,skin safe\n");

printf("$20");

sum = 20.0000;

}

else if(rdlen == 15)

{

printf("ARDUINO UNO BOARD\n");

printf("Open source electronics Prototyping platform\n");

printf("$35");

sum = 35.00;

}

else if(rdlen == 13)

{

printf("SIERRA WIRELESS USB\n");

printf("Made in China\n");

printf("$40");

sum = 40.00;

}

else {

printf("try again");

sum =0.000;

}

// printf(" CodeRead %d: %s", rdlen, buf);

total+= sum;

printf("\nTOTAL = %f\n\n",total);

// for (ii=0;ii<20;ii++) {

// assign[ii]=buf[ii];

// }

}

// printf("the assign variable is:%s\n", assign);

// printf("the bars variable is:%s\n", bars);

/\* else if(rdlen >0 ) /\* display hex \*/

//{ // unsigned char \*p;

// printf("Read %d:", rdlen);

// for (p = buf; rdlen-- > 0; p++)

// printf(" 0x%x", \*p);

// printf("\n");

// {

// buf[rdlen] = 0;

// printf("First Item code %d: \"%s\n", rdlen, buf);

//}

//endif

if (rdlen < 0) {

printf("Error from read: %d: %s\n", rdlen, strerror(errno));

}

/\* repeat read to get full message \*/

} while (1);

}