Udp.c

```
#include <netinet/in_systm.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <netinet/tcp.h>
#include <stdlib.h>
#include <arpa/inet.h>

struct psd_udp {
	struct in_addr src;
	struct in_addr dst;
	unsigned char pad;
	unsigned char proto;
	unsigned short udp_len;
	struct udphdr udp;
};

unsigned short in_cksum(unsigned short *addr, int len)
{
	int nleft=len;

	int sum=0;
	unsigned short *w =addr;
	unsigned short answer=0;

	while(nleft>1)
	{
		sum+=*w++;
		nleft-=2;
	}

	if(nleft==1){
*(unsigned char *)&answer = *(unsigned char *)w;
		sum+=answer;

	}

	sum= (sum>>16) + (sum & 0xFFFFF);
	sum+= sum>>16;
	answer = ~sum;
	return answer;


}
```

```
unsigned short in_cksum_udp(int src, int dst, unsigned short *addr, int len)
{
	struct psd_udp buf;

	memset(&buf, 0, sizeof(buf));
	buf.src.s_addr = src;
	buf.dst.s_addr = dst;
	buf.pad = 0;
	buf.proto = IPPROTO_UDP;
	buf.udp_len = htons(len);
	memcpy(&(buf.udp), addr, len);
	return in_cksum((unsigned short *)&buf, 12 + len);
}



void *run(void *arg)
{
	struct ip ip;
	struct udphdr udp;
	int sd;
	const int on = 1;
	struct sockaddr_in sin;


	u_char *packet;
	packet = (u_char *)malloc(60);


	ip.ip_hl=0x5;
	ip.ip_v=0x4;
	ip.ip_tos=0x0;
	ip.ip_len=60;
	ip.ip_id=htons(12830);
	ip.ip_off=0x0;
	ip.ip_ttl=64;
	ip.ip_p=IPPROTO_UDP;
	ip.ip_sum=0x0;
	ip.ip_src.s_addr=inet_addr("127.0.0.1");
	ip.ip_dst.s_addr= inet_addr("127.0.0.1");

	ip.ip_sum=in_cksum((unsigned short *)&ip , sizeof(ip));
	memcpy(packet, &ip, sizeof(ip));

	udp.uh_sport = htons(45512);
	udp.uh_dport = htons(23);
```

```
	udp.uh_ulen = htons(8);
	udp.uh_sum = in_cksum_udp(ip.ip_src.s_addr, ip.ip_dst.s_addr, (unsigned
short*)&udp, sizeof(udp));
	memcpy(packet + 20, &udp, sizeof(udp));

	//fill IP and UDP here

	if ((sd = socket(AF_INET, SOCK_RAW, IPPROTO_UDP)) < 0) {
		perror("raw socket");
		exit(1);
	}

	if (setsockopt(sd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0) {
		perror("setsockopt");
		exit(1);
	}


	memset(&sin, 0, sizeof(sin));
	sin.sin_family = AF_INET;
	sin.sin_addr.s_addr = ip.ip_dst.s_addr;
	int c=0;
	while(1)
	{
	// c+=1;

	if (sendto(sd, packet, 60, 0, (struct sockaddr *)&sin, sizeof(struct
sockaddr)) < 0)  {
		perror("sendto");
		exit(1);
	}
	}

	}


int main(int argc, char **argv)
{
	run(NULL);
	return 0;
}
```

udp_data .c

```
#include <netinet/in_systm.h>
#include <netinet/in.h>
```

```c
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <netinet/tcp.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <string.h>

struct psd_udp {
	struct in_addr src;
	struct in_addr dst;
	unsigned char pad;
	unsigned char proto;
	unsigned short udp_len;
	struct udphdr udp;
};

unsigned short in_cksum(unsigned short *addr, int len) {
	int nleft = len;
	int sum = 0;
	unsigned short *w = addr;
	unsigned short answer = 0;

	while (nleft > 1) {
	sum += *w++;
	nleft -= 2;
	}

	if (nleft == 1) {
	*(unsigned char *)&answer = *(unsigned char *)w;
	sum += answer;
	}

	sum = (sum >> 16) + (sum & 0xFFFF);
	sum += sum >> 16;
	answer = ~sum;
	return answer;
}

unsigned short in_cksum_udp(int src, int dst, unsigned short *addr, int
len) {
	struct psd_udp buf;

	memset(&buf, 0, sizeof(buf));
```

```
	buf.src.s_addr = src;
	buf.dst.s_addr = dst;
	buf.pad = 0;
	buf.proto = IPPROTO_UDP;
	buf.udp_len = htons(len);
	memcpy(&(buf.udp), addr, len);
	return in_cksum((unsigned short *)&buf, 12 + len);
}

void *run(void *arg) {
	struct ip ip;
	struct udphdr udp;
	int sd;
	const int on = 1;
	struct sockaddr_in sin;

	u_char *packet;
	packet = (u_char *)malloc(60);

	ip.ip_hl = 0x5;
	ip.ip_v = 0x4;
	ip.ip_tos = 0x0;
	ip.ip_len = 60;
	ip.ip_id = htons(12830);
	ip.ip_off = 0x0;
	ip.ip_ttl = 64;
	ip.ip_p = IPPROTO_UDP;
	ip.ip_sum = 0x0;
	ip.ip_src.s_addr = inet_addr("127.0.0.1");
	ip.ip_dst.s_addr = inet_addr("127.0.0.1");

	ip.ip_sum = in_cksum((unsigned short *)&ip, sizeof(ip));
	memcpy(packet, &ip, sizeof(ip));

	udp.uh_sport = htons(45512);
	udp.uh_dport = htons(23);
	udp.uh_ulen = htons(14);  // Length of UDP header + message
	udp.uh_sum = in_cksum_udp(ip.ip_src.s_addr, ip.ip_dst.s_addr,
(unsigned short *)&udp, sizeof(udp));

	// Copy the UDP header
	memcpy(packet + 20, &udp, sizeof(udp));
```

```
	// Add the message "hello" after the UDP header
	const char *message = "hello";
	memcpy(packet + 28, message, strlen(message));

	// fill IP and UDP here

	if ((sd = socket(AF_INET, SOCK_RAW, IPPROTO_UDP)) < 0) {
	perror("raw socket");
	exit(1);
	}

	if (setsockopt(sd, IPPROTO_IP, IP_HDRINCL, &on, sizeof(on)) < 0) {
	perror("setsockopt");
	exit(1);
	}

	memset(&sin, 0, sizeof(sin));
	sin.sin_family = AF_INET;
	sin.sin_addr.s_addr = ip.ip_dst.s_addr;
	int c = 0;
	while (1) {
	// c+=1;

	if (sendto(sd, packet, 60, 0, (struct sockaddr *)&sin, sizeof(struct
sockaddr)) < 0) {
		perror("sendto");
		exit(1);
	}
	}
}

int main(int argc, char **argv) {
	run(NULL);
	return 0;
}
```