# Administrativia

Mark Allman
*mark.allman@case.edu*

EECS 325/425
Oct 17 2018

# Docket

| | This Week | Next Week |
|---|---|---|
| **Mon** | Lecture: Network Layer<br>Return Midterm | Fall Break; No Class<br>No Office Hours<br>Midterm Grades Posted |
| **Tue** | | |
| **Wed** | **Lecture: Network Layer**<br>**Project #3 Due**<br>**Assign Project #4** | Lecture: Network Layer<br>Project #3 Returned … ? |
| **Thu** | | |
| **Fri** | | |

# Project 3

```
HTTP/1.1 400 Malformed Request
HTTP/1.1 501 Protocol Not Implemented
HTTP/1.1 405 Unsupported Method
HTTP/1.1 200 Server Shutting Down
HTTP/1.1 403 Operation Forbidden
HTTP/1.1 406 Invalid Filename
HTTP/1.1 200 OK
HTTP/1.1 404 File Not Found
```

# Project 3

- Questions?

# Network Layer

- Questions?

# Project 4

# Project 4

- Packet trace analysis

# Project 4

- Packet trace analysis

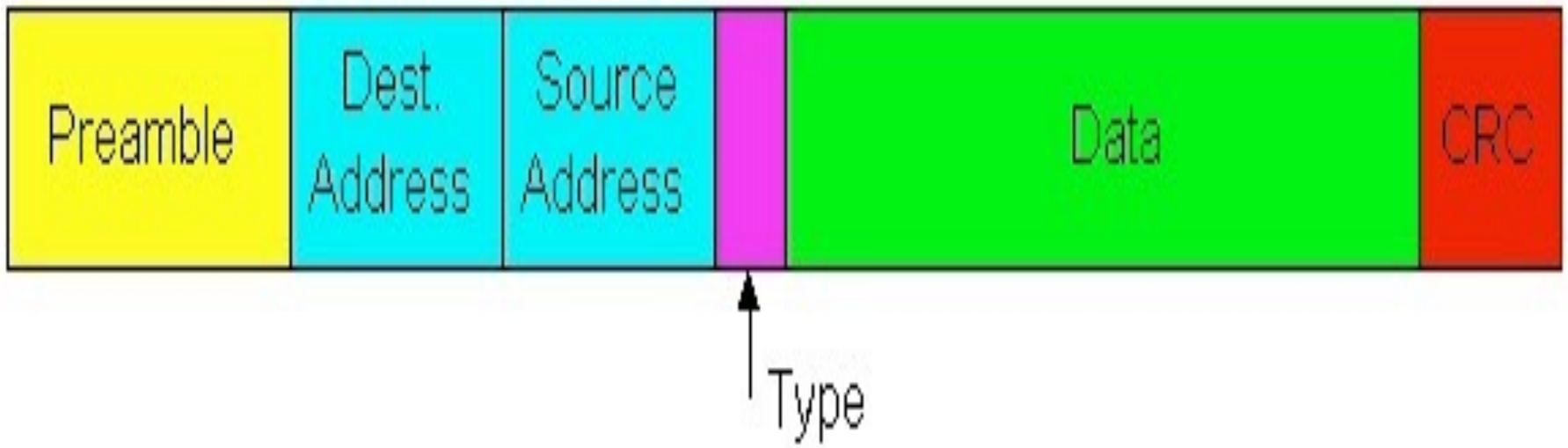
- Packet traces are fraught with ethical issues

# Project 4

- Packet trace analysis

- Packet traces are fraught with ethical issues
  - The packet traces you will use use two coping strategies
    - the payload has been stripped
    - the headers have been anonymized

# Project 4



# Project 4

- The ‘od’ tool may well be useful to pick through binary files to ensure you’re getting the correct values
  - especially for small trace files
  - see the Project Tip session slides
  - “man od”

# Project 4

```
struct pkt_info pinfo;

fd = open (…);
while (next_packet (fd,&pinfo))
{
    /* do something with packet
       in pinfo */
}
close (fd);
```

# Project 4

# Project 4

```
#define MAX_PKT_SIZE 1600

/* record of information nabout the current packet */
struct pkt_info
{
    unsigned short caplen;
    double now;
    unsigned char pkt [MAX_PKT_SIZE];
    struct ether_header *ethh;  /* ptr to ethernet header, if fully present */
    struct iphdr *iph;          /* ptr to IP header, if fully present */
    struct tcphdr *tcph;        /* ptr to TCP header, if fully present */
    struct udphdr *udph;        /* ptr to UDP header, if fully present */
};
```

# Project 4

```
#define MAX_PKT_SIZE 1600

/* record of information nabout the current packet */
struct pkt_info
{
    unsigned short caplen;
    double now;
    unsigned char pkt [MAX_PKT_SIZE];
    struct ether_header *ethh;  /* ptr to ethernet header, if fully present */
    struct iphdr *iph;          /* ptr to IP header, if fully present */
    struct tcphdr *tcph;        /* ptr to TCP header, if fully present */
    struct udphdr *udph;        /* ptr to UDP header, if fully present */
};
```

pkt: 

# Project 4

```
#define MAX_PKT_SIZE 1600

/* record of information nabout the current packet */
struct pkt_info
{
    unsigned short caplen;
    double now;
    unsigned char pkt [MAX_PKT_SIZE];
    struct ether_header *ethh;  /* ptr to ethernet header, if fully present */
    struct iphdr *iph;          /* ptr to IP header, if fully present */
    struct tcphdr *tcph;        /* ptr to TCP header, if fully present */
    struct udphdr *udph;        /* ptr to UDP header, if fully present */
};
```

# Project 4

```
#define MAX_PKT_SIZE 1600

/* record of information nabout the current packet */
struct pkt_info
{
    unsigned short caplen;
    double now;
    unsigned char pkt [MAX_PKT_SIZE];
    struct ether_header *ethh;  /* ptr to ethernet header, if fully present */
    struct iphdr *iph;          /* ptr to IP header, if fully present */
    struct tcphdr *tcph;        /* ptr to TCP header, if fully present */
    struct udphdr *udph;        /* ptr to UDP header, if fully present */
};
```

# Project 4

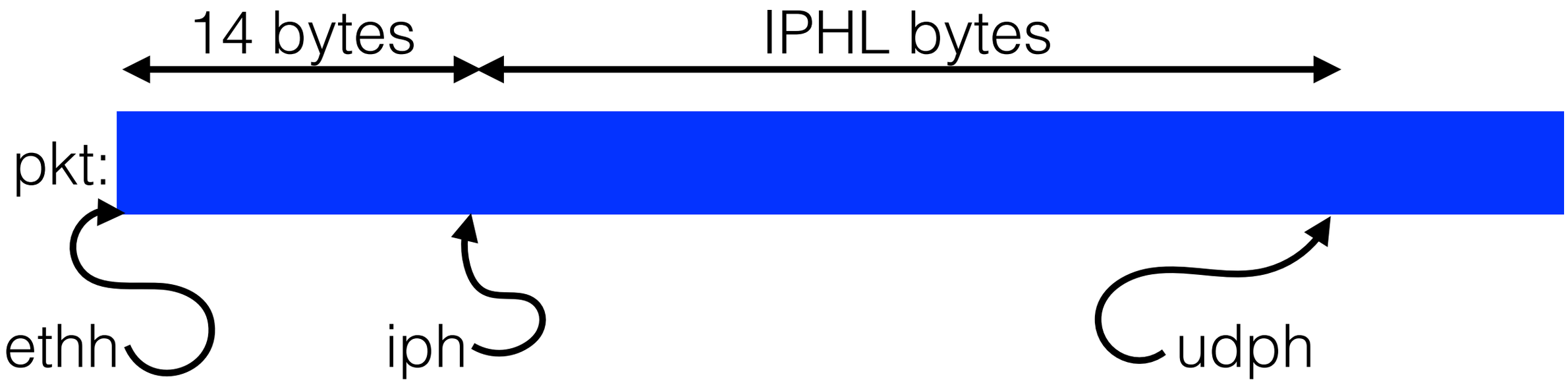
```
#define MAX_PKT_SIZE 1600

/* record of information nabout the current packet */
struct pkt_info
{
    unsigned short caplen;
    double now;
    unsigned char pkt [MAX_PKT_SIZE];
    struct ether_header *ethh;  /* ptr to ethernet header, if fully present */
    struct iphdr *iph;          /* ptr to IP header, if fully present */
    struct tcphdr *tcph;        /* ptr to TCP header, if fully present */
    struct udphdr *udph;        /* ptr to UDP header, if fully present */
};
```

# Project 4

```
#define MAX_PKT_SIZE 1600

/* record of information nabout the current packet */
struct pkt_info
{
    unsigned short caplen;
    double now;
    unsigned char pkt [MAX_PKT_SIZE];
    struct ether_header *ethh;  /* ptr to ethernet header, if fully present */
    struct iphdr *iph;          /* ptr to IP header, if fully present */
    struct tcphdr *tcph;        /* ptr to TCP header, if fully present */
    struct udphdr *udph;        /* ptr to UDP header, if fully present */
};
```

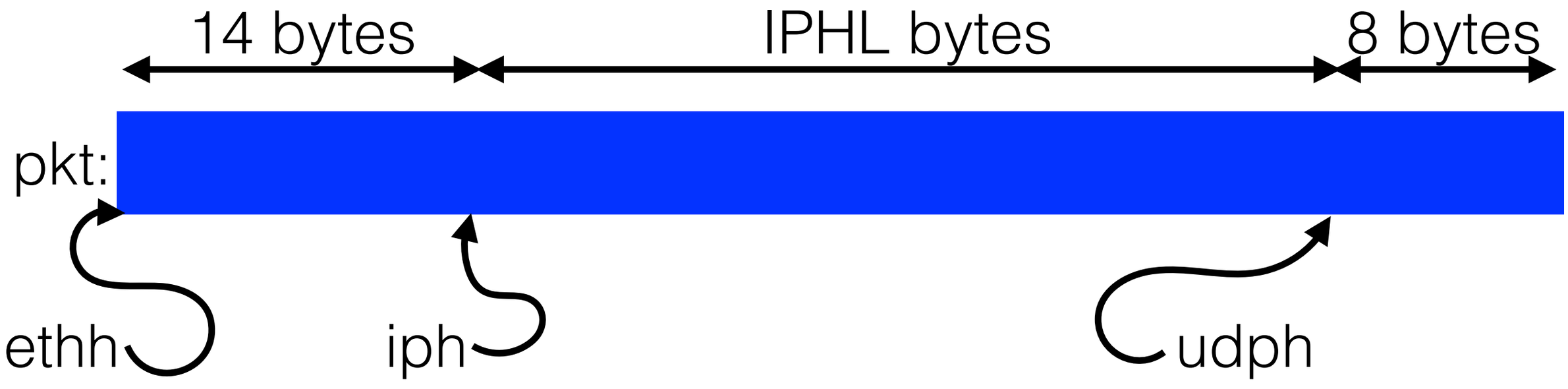
# Project 4

```
#define MAX_PKT_SIZE 1600

/* record of information nabout the current packet */
struct pkt_info
{
    unsigned short caplen;
    double now;
    unsigned char pkt [MAX_PKT_SIZE];
    struct ether_header *ethh;  /* ptr to ethernet header, if fully present */
    struct iphdr *iph;          /* ptr to IP header, if fully present */
    struct tcphdr *tcph;        /* ptr to TCP header, if fully present */
    struct udphdr *udph;        /* ptr to UDP header, if fully present */
};
```



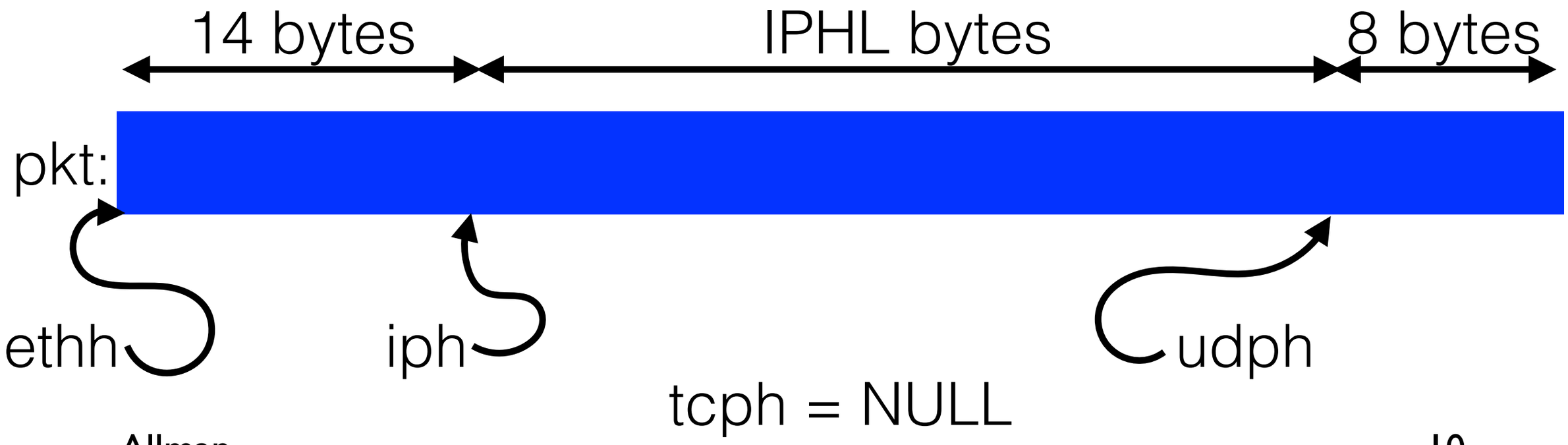
# Project 4

```
#define MAX_PKT_SIZE 1600

/* record of information nabout the current packet */
struct pkt_info
{
    unsigned short caplen;
    double now;
    unsigned char pkt [MAX_PKT_SIZE];
    struct ether_header *ethh;  /* ptr to ethernet header, if fully present */
    struct iphdr *iph;          /* ptr to IP header, if fully present */
    struct tcphdr *tcph;        /* ptr to TCP header, if fully present */
    struct udphdr *udph;        /* ptr to UDP header, if fully present */
};
```



# Project 4

```
#define MAX_PKT_SIZE 1600

/* record of information nabout the current packet */
struct pkt_info
{
    unsigned short caplen;
    double now;
    unsigned char pkt [MAX_PKT_SIZE];
    struct ether_header *ethh;  /* ptr to ethernet header, if fully present */
    struct iphdr *iph;          /* ptr to IP header, if fully present */
    struct tcphdr *tcph;        /* ptr to TCP header, if fully present */
    struct udphdr *udph;        /* ptr to UDP header, if fully present */
};
```



# Project 4

```
/* meta information, using same layout as trace file */
struct meta_info
{
    unsigned short caplen;
    unsigned short ignored;
    unsigned int secs;
    unsigned int usecs;
};
```

# Project 4

```
struct pkt_info pinfo;

fd = open (…);
while (next_packet (fd,&pinfo))
{
    /* do something with packet
       in pinfo */
}
close (fd);
```

# Project 4

# Project 4

```
unsigned short next_packet (int fd, struct pkt_info *pinfo)
{
    struct meta_info meta;
    int bytes_read;
```

# Project 4

```
unsigned short next_packet (int fd, struct pkt_info *pinfo)
{
    struct meta_info meta;
    int bytes_read;

    memset (pinfo,0x0,sizeof (struct pkt_info));
    memset (&meta,0x0,sizeof (struct meta_info));
```

# Project 4

```
unsigned short next_packet (int fd, struct pkt_info *pinfo)
{
    struct meta_info meta;
    int bytes_read;

    memset (pinfo,0x0,sizeof (struct pkt_info));
    memset (&meta,0x0,sizeof (struct meta_info));

    bytes_read = read (fd,&meta,sizeof (meta));
```

# Project 4

```
unsigned short next_packet (int fd, struct pkt_info *pinfo)
{
    struct meta_info meta;
    int bytes_read;

    memset (pinfo,0x0,sizeof (struct pkt_info));
    memset (&meta,0x0,sizeof (struct meta_info));

    bytes_read = read (fd,&meta,sizeof (meta));
    if (bytes_read == 0)
     return (0);
```

# Project 4

```
unsigned short next_packet (int fd, struct pkt_info *pinfo)
{
    struct meta_info meta;
    int bytes_read;

    memset (pinfo,0x0,sizeof (struct pkt_info));
    memset (&meta,0x0,sizeof (struct meta_info));

    bytes_read = read (fd,&meta,sizeof (meta));
    if (bytes_read == 0)
     return (0);
    if (bytes_read < sizeof (meta))
     errexit ("cannot read meta information");
```

# Project 4

```
unsigned short next_packet (int fd, struct pkt_info *pinfo)
{
    struct meta_info meta;
    int bytes_read;

    memset (pinfo,0x0,sizeof (struct pkt_info));
    memset (&meta,0x0,sizeof (struct meta_info));

    bytes_read = read (fd,&meta,sizeof (meta));
    if (bytes_read == 0)
     return (0);
    if (bytes_read < sizeof (meta))
     errexit ("cannot read meta information");
    pinfo->caplen = ntohs (meta.caplen);
```

# Project 4

```
unsigned short next_packet (int fd, struct pkt_info *pinfo)
{
    struct meta_info meta;
    int bytes_read;

    memset (pinfo,0x0,sizeof (struct pkt_info));
    memset (&meta,0x0,sizeof (struct meta_info));

    bytes_read = read (fd,&meta,sizeof (meta));
    if (bytes_read == 0)
     return (0);
    if (bytes_read < sizeof (meta))
     errexit ("cannot read meta information");
    pinfo->caplen = ntohs (meta.caplen);
    /* set pinfo->now from meta information */
```

# Project 4

```
unsigned short next_packet (int fd, struct pkt_info *pinfo)
{
    struct meta_info meta;
    int bytes_read;

    memset (pinfo,0x0,sizeof (struct pkt_info));
    memset (&meta,0x0,sizeof (struct meta_info));

    bytes_read = read (fd,&meta,sizeof (meta));
    if (bytes_read == 0)
        return (0);
    if (bytes_read < sizeof (meta))
        errexit ("cannot read meta information");
    pinfo->caplen = ntohs (meta.caplen);
    /* set pinfo->now from meta information */
    if (pinfo->caplen == 0)
        return (1);
```

# Project 4

```
unsigned short next_packet (int fd, struct pkt_info *pinfo)
{
    struct meta_info meta;
    int bytes_read;

    memset (pinfo,0x0,sizeof (struct pkt_info));
    memset (&meta,0x0,sizeof (struct meta_info));

    bytes_read = read (fd,&meta,sizeof (meta));
    if (bytes_read == 0)
        return (0);
    if (bytes_read < sizeof (meta))
        errexit ("cannot read meta information");
    pinfo->caplen = ntohs (meta.caplen);
    /* set pinfo->now from meta information */
    if (pinfo->caplen == 0)
        return (1);
    if (pinfo->caplen > MAX_PKT_SIZE)
        errexit ("packet too big");
    [ … ]
```

# Project 4

# Project 4

[ … ]

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
```

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
if (bytes_read < pinfo->caplen)
    errexit ("unexpected end of file encountered");
```

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
if (bytes_read < pinfo->caplen)
    errexit ("unexpected end of file encountered");
if (bytes_read < sizeof (struct ether_header))
    return (1);
```

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
if (bytes_read < pinfo->caplen)
    errexit ("unexpected end of file encountered");
if (bytes_read < sizeof (struct ether_header))
    return (1);
pinfo->ethh = (struct ether_header *)pinfo->pkt;
```

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
if (bytes_read < pinfo->caplen)
    errexit ("unexpected end of file encountered");
if (bytes_read < sizeof (struct ether_header))
    return (1);
pinfo->ethh = (struct ether_header *)pinfo->pkt;
pinfo->ethh->ether_type = ntohs (pinfo->ethh->ether_type);
```

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
if (bytes_read < pinfo->caplen)
    errexit ("unexpected end of file encountered");
if (bytes_read < sizeof (struct ether_header))
    return (1);
pinfo->ethh = (struct ether_header *)pinfo->pkt;
pinfo->ethh->ether_type = ntohs (pinfo->ethh->ether_type);
/* possibly set pinfo->iph and handle byte order */
```

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
if (bytes_read < pinfo->caplen)
    errexit ("unexpected end of file encountered");
if (bytes_read < sizeof (struct ether_header))
    return (1);
pinfo->ethh = (struct ether_header *)pinfo->pkt;
pinfo->ethh->ether_type = ntohs (pinfo->ethh->ether_type);
/* possibly set pinfo->iph and handle byte order */
/* possibly set pinfo->udph and handle byte order */
```

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
if (bytes_read < pinfo->caplen)
    errexit ("unexpected end of file encountered");
if (bytes_read < sizeof (struct ether_header))
    return (1);
pinfo->ethh = (struct ether_header *)pinfo->pkt;
pinfo->ethh->ether_type = ntohs (pinfo->ethh->ether_type);
/* possibly set pinfo->iph and handle byte order */
/* possibly set pinfo->udph and handle byte order */
/* possibly set pinfo->tcph and handle byte order */
```

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
if (bytes_read < pinfo->caplen)
    errexit ("unexpected end of file encountered");
if (bytes_read < sizeof (struct ether_header))
    return (1);
pinfo->ethh = (struct ether_header *)pinfo->pkt;
    pinfo->ethh->ether_type = ntohs (pinfo->ethh->ether_type);
    /* possibly set pinfo->iph and handle byte order */
    /* possibly set pinfo->udph and handle byte order */
    /* possibly set pinfo->tcph and handle byte order */
    return (1);
}
```

# Project 4

```
[ … ]
bytes_read = read (fd,pinfo->pkt,pinfo->caplen);
if (bytes_read < pinfo->caplen)
    errexit ("unexpected end of file encountered");
if (bytes_read < sizeof (struct ether_header))
    return (1);
pinfo->ethh = (struct ether_header *)pinfo->pkt;
    pinfo->ethh->ether_type = ntohs (pinfo->ethh->ether_type);
    /* possibly set pinfo->iph and handle byte order */
    /* possibly set pinfo->udph and handle byte order */
    /* possibly set pinfo->tcph and handle byte order */
    return (1);
}
```

# Project 4

```
struct pkt_info pinfo;

fd = open (…);
while (next_packet (fd,&pinfo))
{
    /* do something with packet
       in pinfo */
}
close (fd);
```