Sheet 2

socket structure

struct socket

```
struct socket
{
```

The socket state can be: SS_FREE, SS_UNCONNECTED, SS_CONNECTING, SS_CONNECTED, SS_DISCONNECTING

```
    socket_state           state;

    unsigned long          flags;
```

Defined below.

```
    struct proto_ops      *ops;
```

In unix, each file is described by an inode. In Linux, there is also an inode for each BSD socket. Hence:

`inode` holds a reference to the corresponding inode for this socket

`file` holds a reference to the primary file structure associated with this socket

```
    struct inode          *inode;
    struct file           *file;           /* File back pointer for gc    */
```

`fasync_list` is used when processes have chosen asynchronous handling of this 'file'

```
    struct fasync_struct  *fasync_list;    /* Asynchronous wake up list   */
```

This is very important, as it contains most of the useful state associated with a socket. See later

```
23      struct sock             *sk;
24
```
Not used by sockets in AF_INET
```
26      wait_queue_head_t       wait;
27
```
The type is SOCK_STREAM, SOCK_DGRAM, SOCK_RAW
```
29      short                   type;
30      unsigned char           passcred;
31  };
32
33
```

struct proto_ops

These are initialised for during creation of struct sock in e.g. net/ipv4/af_inet.c::inet_create which pulls them out of a local table, net/ipv4/af_inet.c::inetsw, which is itself initialised by net/ipv4/af_inet.c::inet_register_protosw which is called from net/ipv4/af_inet.c::inet_init, which uses the static array net/ipv4/af_inet.c::inetsw_array

```
struct proto_ops {
  int    family;

  int    (*release)   (struct socket *sock);
  int    (*bind)      (struct socket *sock, struct sockaddr *umyaddr,
                           int sockaddr_len);
  int    (*connect)   (struct socket *sock, struct sockaddr *uservaddr,
                           int sockaddr_len, int flags);
  int    (*socketpair)(struct socket *sock1, struct socket *sock2);
  int    (*accept)    (struct socket *sock, struct socket *newsock, int flags);
  int    (*getname)   (struct socket *sock, struct sockaddr *uaddr,
                           int *usockaddr_len, int peer);
  unsigned int (*poll)(struct file *file, struct socket *sock,
                           struct poll_table_struct *wait);
  int    (*ioctl)     (struct socket *sock, unsigned int cmd, unsigned long arg);
  int    (*listen)    (struct socket *sock, int len);
  int    (*shutdown)  (struct socket *sock, int flags);
  int    (*setsockopt)(struct socket *sock, int level, int optname,
```

```
57                              char *optval, int optlen);
58    int    (*getsockopt)(struct socket *sock, int level, int optname,
59                              char *optval, int *optlen);
60    int    (*sendmsg)    (struct socket *sock, struct msghdr *m, int total_len, struct
61                              scm_cookie *scm);
62    int    (*recvmsg)    (struct socket *sock, struct msghdr *m, int total_len,
63                              int flags, struct scm_cookie *scm);
64    int    (*mmap)       (struct file *file, struct socket *sock, struct
65                              vm_area_struct * vma);
66    ssize_t (*sendpage) (struct socket *sock, struct page *page, int offset,
67                              size_t size, int flags);
68  };
69
```