

```

#define BLEN(s) ((s)->wp - (s)->rp)
#define BALLOC(s) ((s)->lim - (s)->base)
    
```

<pre> Chan Ref; Chan* next; Chan* link; vlong offset; vlong devoffset; ushort type; ulong dev; ushort mode; ushort flag; Qid qid; int fid; ulong iounit; Mhead* umh; Chan* umc; QLock umqlock; int uri; int dri; uchar* dirrock; int nrock; int mrock; QLock rockqlock; int ismtp; ulong mountid; Mntcache*mcp; Mnt* mux; union { void* aux; Qid pgrp; ulong mid; }; Chan* mchan; Qid mqid; Path * path; </pre>	<p>the Lock in this Ref is also Chan's lock allocation</p> <p>in fd in underlying device; see read</p> <p>read/write</p> <p>for devmnt chunk size for i/o; 0==default</p> <p>mount point that derived Chan; used in unionread</p> <p>channel in union; held for union read serialize unionreads */</p> <p>union read index devdirread index directory entry rock for translations</p> <p>Mount cache pointer</p> <p>Mnt for clients using me for messages</p> <p>for #p/notepg for ns in devproc</p> <p>channel to mounted server qid of root of mount point</p>
---	--

<pre> Path Ref; char* s; Chan** mtpt; int len; int alen; int mlen; int malen; </pre>	<p>strlen(s) allocated length num of path elements allocated len of mtpt</p>
--	--

```

- Dev
int    dc;    //デバイスタイプ cf. major nr.
char*  name;

void    (*reset)(void);
void    (*init)(void);
void    (*shutdown)(void);
Chan*   (*attach)(char*);
Walkqid*(*walk)(Chan*, Chan*, char**, int);
int     (*stat)(Chan*, uchar*, int);
Chan*   (*open)(Chan*, int);
void    (*create)(Chan*, char*, int, ulong);
void    (*close)(Chan*);
long    (*read)(Chan*, void*, long, vlong);
Block*  (*bread)(Chan*, long, ulong);
long    (*write)(Chan*, void*, long, vlong);
long    (*bwrite)(Chan*, Block*, ulong);
void    (*remove)(Chan*);
int     (*wstat)(Chan*, uchar*, int);
void    (*power)(int);
int     (*config)(int, char*, DevConf*);

```

デバイス=カーネルオブジェクト

```

- Dirtab
char    name[KNAMELEN];
Qid     qid;
vlong   length;
long    perm;

```

```

- Walkqid
Chan    *clone;
int     nqid;
Qid     qid[1];

```

```

- Mntwalk
int     cddone;
ulong   id;
Mhead*  mh;
Mount*  cm;

```

```

- Mount
ulong   mountid;
Mount*  next;
Mhead*  head;
Mount*  copy;
Mount*  order;
Chan*   to;
int     mflag;
char    *spec;

```

channel replacing channel */

```

- Mhead
Ref;
RWlock  lock;
Chan*   from;
Mount*  mount;
Mhead*  hash;

```

channel mounted upon */
what's mounted upon it */
Hash chain */

<pre> Mnt Lock; Chan *c; Proc *rip; Mntrpc *queue; ulong id; Mnt *list; int flags; int msize; char *version; Queue *q; </pre>	<pre> Channel to file service */ Reader in progress */ Queue of pending requests on this channel */ Multiplexer id for channel check */ Free list */ cache */ data + IOHDRSZ */ 9P version */ input queue */ </pre>
---	---

<pre> Ref Lock; long ref; </pre>

<pre> Rendez Lock; Proc *p; </pre>

<pre> QLock Lock use; Proc *head; Proc *tail; int locked; </pre>	<pre> to access Qlock structure next process waiting for object last process waiting for object flag */ </pre>
---	--

<pre> RWlock Lock use; Proc *head; Proc *tail; ulong wpc; Proc *wproc; int readers; int writer; </pre>	<pre> list of waiting processes pc of writer writing proc number of readers number of writers </pre>
--	---

<pre> Talarm Lock; Proc *list; </pre>
--

<pre> Alarms QLock; Proc *head; </pre>

<pre> Sargs ulong args [MAXSYSARG]; </pre>

```

SDperm
char*  name;
char*  user;
ulong  perm;

```

```

SDpart
ulong  start;
ulong  end;
SDperm;
int    valid;
ulong  vers;

```

```

SDunit
SDev*  dev;
int    subno;
uchar  inquiry[256];
SDperm;

QLock  ctl;
ulong  sectors;
ulong  secsize;
SDpart* part;
int    npart;
ulong  vers;
SDperm ctlperm;

QLock  raw;
ulong  rawinuse;
int    state;
SDreq* req;
SDperm rawperm;

```

```

SDev
Ref    r;
SDifc* ifc;
void*  ctlr;
int    idno;
char*  name;
SDev*  next;

QLock;
int    enabled;
int    nunit;
QLock  unitlock;
int*   unitflg;
SDunit**unit;

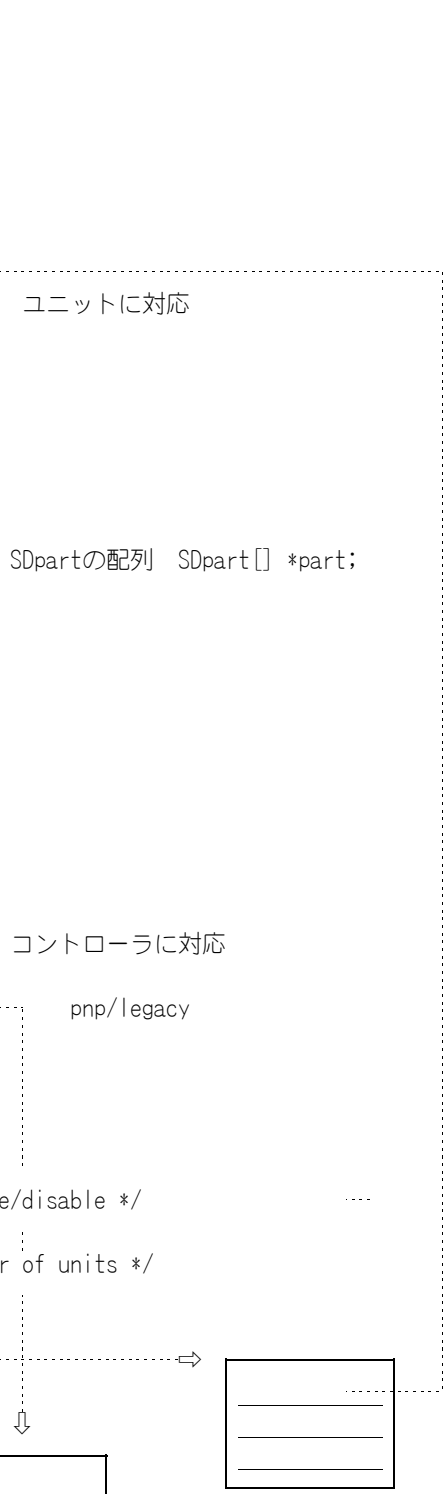
```

```

SDifc
char*  name;

SDev*  (*pnp) (void);
SDev*  (*legacy) (int, int);
SDev*  (*id) (SDev*);
int    (*enable) (SDev*);

```



```
int (*disable) (SDev*);

int (*verify) (SDunit*);
int (*online) (SDunit*);
int (*rio) (SDreq*);
int (*rctl) (SDunit*, char*, int);
int (*wctl) (SDunit*, Cmdbuf*);
long (*bio) (SDunit*, int, int,
            void*, long, long);
SDev* (*probe) (DevConf*);
void (*clear) (SDev*);
char* (*stat) (SDev*, char*, char*);
```

```
----- Dreq -----
SDunit* unit;
int lun;
int write;
uchar cmd[16];
int clen;
void* data;
int dlen;

int flags;

int status;
long rlen;
uchar sense[256];
```

```
* one per conversation directory
```

```

- Conv
QLock;
int x; /* conversation index */
Proto* p;
int restricted; /* remote port is restricted */
uint ttl; /* max time to live */
uint tos; /* type of service */
int ignoreadvise; /* don't terminate connection on icmp errors */
uchar ipversion;
uchar laddr[IPAddrLen]; /* local IP address */
uchar raddr[IPAddrLen]; /* remote IP address */
ushort lport; /* local port number */
ushort rport; /* remote port number */
char *owner; /* protections */
int perm;
int inuse; /* opens of listen/data/ctl */
int length;
int state;
/* udp specific */
int headers; /* data src/dst headers in udp */
int reliable; /* true if reliable udp */
Conv* incall; /* calls waiting to be listened for */
Conv* next;
Queue* rq; /* queued data waiting to be read */
Queue* wq; /* queued data waiting to be written */
Queue* eq; /* returned error packets */
Queue* sq; /* snooping queue */
Ref snoopers; /* number of processes with snoop open */
QLock car;
Rendez cr;
char cerr[ERRMAX];
QLock listenq;
Rendez listenr;
lpmulti *multi; /* multicast bindings for this interface */
void* ptcl; /* protocol specific stuff */
Route *r; /* last route used */
ulong rgen; /* routetable generation for *r */

```

```

- Medium
char *name;
int hsize; /* medium header size */
int mintu; /* default min mtu */
int maxtu; /* default max mtu */
int maclen; /* mac address length */
void (*bind)(lpifc*, int, char**);
void (*unbind)(lpifc*);
void (*bwrite)(lpifc *ifc, Block *b, int version, uchar *ip);
/* for arming interfaces to receive multicast */
void (*addmulti)(lpifc *ifc, uchar *a, uchar *ia);
void (*remmulti)(lpifc *ifc, uchar *a, uchar *ia);
/* process packets written to 'data' */
void (*pktin)(Fs *f, lpifc *ifc, Block *bp);
/* routes for router boards */
void (*addroute)(lpifc *ifc, int, uchar*, uchar*, uchar*, int);
void (*remroute)(lpifc *ifc, int, uchar*, uchar*);
void (*flushroutes)(lpifc *ifc);

```

```

/* for routing multicast groups */
void (*joinmulti)(lplifc *lplifc, uchar *a, uchar *ia);
void (*leavemulti)(lplifc *lplifc, uchar *a, uchar *ia);
/* address resolution */
void (*ares)(Fs*, int, uchar*, uchar*, int, int); /* resolve */
void (*areg)(lplifc*, uchar*); /* register */
/* v6 address generation */
void (*pref2addr)(uchar *pref, uchar *ea);
int unbindonclose; /* if non-zero, unbind on last close */

```

<pre> lplifc uchar local[IPAddrLen]; uchar mask[IPAddrLen]; uchar remote[IPAddrLen]; uchar net[IPAddrLen]; uchar tentative; /* =1 => v6 dup disc on, =0 => confirmed unique */ uchar onlink; /* =1 => onlink, =0 offlink. */ uchar autoflag; /* v6 autonomous flag */ long validlt; /* v6 valid lifetime */ long preflt; /* v6 preferred lifetime */ long origint; /* time when addr was added */ lplink *link; /* addresses linked to this lifc */ lplifc *next; </pre>	logical interface associated with a physical one
---	--

<pre> lplink lpself *self; lplifc *lifc; lplink *selflink; lplink *lifclink; ulong expire; lplink *next; int ref; </pre>	binding twixt lpself and lplifc next link for this local address */ next link for this ifc */ free list */
--	---

<pre> Routerparams int mflag; int oflag; int maxraint; int minraint; int linkmtu; int reachtime; int rxmitra; int ttl; int routerlt; </pre>	default values, one per stack
---	-------------------------------

<pre> Hostparams int rxmithost; </pre>	
--	--

<pre> lplifc RWlock; Conv *conv; char dev[64]; Medium *m; int maxtu; int mintu; int mbps; void *arg; </pre>	link to its conversation structure */ device we're attached to */ Media pointer */ Maximum transfer unit */ Minimum tranfer unit */ megabits per second */ medium specific */
---	---

int	reassemble;	reassemble IP packets before forwarding */
/* these are used so that we can unbind on the fly */		
Lock	idlock;	
uchar	ifcid;	incremented each 'bind/unbind/add/remove' */
int	ref;	number of proc's using this ipifc */
Rendez	wait;	where unbinder waits for ref == 0 */
int	unbinding;	
uchar	mac[MAClen];	MAC address */
lplifc	*lifc;	logical interfaces on this physical one */
ulong	in, out;	message statistics */
ulong	inerr, outerr;	... */
uchar	sendra6;	== 1 => send router advs on this ifc */
uchar	recvra6;	== 1 => recv router advs on this ifc */
Routerparams	rp;	router parameters as in RFC 2461, pp.40--43. used only if node is router */

lpmulti		one per multicast-lifc pair used by a Conv
uchar	ma[IPaddrlen];	
uchar	ia[IPaddrlen];	
lpmulti	*next;	

lphash		hash table for 2 ip addresses + 2 ports
lphash	*next;	
Conv	*c;	
int	match;	

lph	
Lock;	
lphash	*tab[Niph];

Proto		one per multiplexed protocol
QLock;		
char*	name;	protocol name */
int	x;	protocol index */
int	ipproto;	ip protocol type */
char*	(*connect) (Conv*, char**, int);	
char*	(*announce) (Conv*, char**, int);	
char*	(*bind) (Conv*, char**, int);	
int	(*state) (Conv*, char*, int);	
void	(*create) (Conv*);	
void	(*close) (Conv*);	
void	(*rcv) (Proto*, lpiifc*, Block*);	
char*	(*ctl) (Conv*, char**, int);	
void	(*advise) (Proto*, Block*, char*);	
int	(*stats) (Proto*, char*, int);	
int	(*local) (Conv*, char*, int);	
int	(*remote) (Conv*, char*, int);	
int	(*inuse) (Conv*);	
int	(*gc) (Proto*);	returns true if any conversations are freed */
Fs	*f;	file system this proto is part of */
Conv	**conv;	array of conversations */
int	ptclsize;	size of per protocol ctl block */
int	nc;	number of conversations */
int	ac;	

```

Qid    qid;          /* qid for protocol directory */
ushort nextport;
ushort nextrport;
void   *priv;

```

```

Fs
RWlock;
int    dev;
int    np;
Proto* p[Maxproto+1]; /* list of supported protocols */
Proto* t2p[256];      /* vector of all protocols */
Proto* ipifc;        /* kludge for ipifcremroute & ipifcaddroute */
Proto* ipmux;        /* kludge for finding an ip multiplexor */
IP     *ip;
Ipselftab *self;
Arp    *arp;
v6params *v6p;
Route  *v4root[1<<Lroot]; /* v4 routing forest */
Route  *v6root[1<<Lroot]; /* v6 routing forest */
Route  *queue;         /* used as temp when reinjecting routes */
Netlog *alog;
char   ndb[1024];     /* an ndb entry for this interface */
int    ndbvers;
long   ndbmtime;

```

one per IP protocol stack

/* one per default router known to host */

```

v6router
uchar  inuse;
Ipifc *ifc;
int    ifcid;
uchar  routeraddr[IPaddrlen];
long   ltorigin;
Routerparams rp;

```

```

v6params
Routerparams rp; /* v6 params, one copy per node now */
Hostparams  hp;
v6router    v6rlist[3]; /* max 3 default routers, currently */
int         cdrouter; /* uses only v6rlist[cdrouter] if cdrouter >= 0. */

```

```

Routewalk
int    o;
int    h;
char*  p;
char*  e;
void*  state;
void   (*walk)(Route*, Routewalk*);

```

```

RouteTree
Route* right;
Route* left;
Route* mid;
uchar  depth;
uchar  type;
uchar  ifcid; /* must match ifc->id */

```

```

Ipifc  *ifc;
char   tag[4];
int    ref;

```

```

V4route
ulong  address;
ulong  endaddress;
uchar  gate[IPv4addrlen];

```

```

V6route
ulong  address[IP1len];
ulong  endaddress[IP1len];
uchar  gate[IPaddrlen];

```

```

Route
RouteTree;
union {
    V6route v6;
    V4route v4;
}

```

```

IPaux
char   *owner;
char   tag[4];

```

the user that did the attach */

```

Arpent
uchar  ip[IPaddrlen];
uchar  mac[MAC1en];
Medium *type;
Arpent* hash;
Block* hold;
Block* last;
uint   ctime;
uint   utime;
uchar  state;
Arpent *nextxt;
uint   rtime;
uchar  rxtsrem;
Ipifc  *ifc;
uchar  ifcid;

```

media type */

time entry was created or refreshed */

time entry was last used */

re-transmit chain */

time for next retransmission */

must match ifc->id */