

kukumouse的专栏

目录视图

摘要视图

RSS 订阅

个人资料



kukumouse

访问：17249次

积分：295

等级：BLOG 2

排名：千里之外

原创：11篇

转载：2篇

译文：0篇

评论：1条

文章搜索

文章分类

C/C++编程 (1)

Linux (2)

voip知识 (4)

vxworks (2)

工作点滴 (1)

网络编程 (1)

行业动态 (0)

文章存档

2008年05月 (1)

2008年04月 (12)

阅读排行

字节序有大端和小端之分 (3467)

边缘会话控制器(SBC) (1550)

ICE实现穿透 (1519)

可变参数宏 (1340)

vxworks常用数据结构 (1242)

DTMF的传输 (1190)

vxworks异常分析 (1103)

如何在Linux下实现定时器 (459)

voip电话技术 (458)

vxworks常用数据结构

标签：数据结构 input output 硬件驱动 list sockets

2008-04-09 15:52 1242人阅读 评论(1) 收藏 举报

分类：vxworks (1)

版权声明：本文为博主原创文章，未经博主允许不得转载。

一: 基本数据结构

typedef struct node/* Node of a linked list. */
{
 struct node *next;/* Points at the next node in the list */
 struct node *previous;/* Points at the previous node in the list */
} NODE;

typedef struct/* Header for a linked list. */
{
 NODE node;/* Header list node */
 int count;/* Number of nodes in list */
} LIST;

typedef struct dlnode/* Node of a linked list. */
{
 struct dlnode *next;/* Points at the next node in the list */
 struct dlnode *previous;/* Points at the previous node in the list */
} DL_NODE;

typedef struct/* Header for a linked list. */
{
 DL_NODE *head;/* header of list */
 DL_NODE *tail;/* tail of list */
} DL_LIST;

二: 网络相关

1.LIST hostList
整个系统的host, hostAdd()产生. hostShow()产看.其中的表项为:
typedef struct
{
 NODE node; /*节点链*/
 HOSTNAME hostName;/*名称*/
 struct in_addr netAddr;/*网络地址*/
} HOSTENTRY;
char targetName [HOSTNAME_LEN];/* name for this target machine */

http://blog.csdn.net/kukumouse/article/details/2270375

1/6

静态库和动态库	(407)
评论排行	
vxworks常用数据结构	(1)
ICE实现穿透	(0)
如何在Linux下实现定时器	(0)
静态库和动态库	(0)
vxworks异常分析	(0)
windows命令集	(0)
字节序有大端和小端之分	(0)
边缘会话控制器(SBC)	(0)
voip电话技术	(0)
DTMF的传输	(0)
推荐文章	
* 致JavaScript也将征服的物联网世界	
* 从苏宁电器到卡巴斯基：难忘的三年硕士时光	
* 作为一名基层管理者如何利用情商管理自己和团队（一）	
* Android CircleImageView圆形ImageView	
* 高质量代码的命名法则	
最新评论	
vxworks常用数据结构匿名用户:	

2.LIST endList

整个系统已加载的end, muxDevLoad()产生,endFindByName()查看.其中表项为:

```
typedef struct end_tbl_row
{
    NODE node;           /* Needed by our list processing library. */
    char name[END_NAME_MAX];/* Name of the devices in row. (ln, etc.) */
    LIST units;          /* List of devices, i.e. 0, 1, etc.. */
} END_TBL_ROW;
```

units链表中包含的是真正的END_OBJ.

3.END_TBL_ENTRY endDevTbl[]

整个系统要加载的end,加载后便进入endList.其中表项为:

```
typedef struct end_tbl_entry
{
    int unit;             /* This device's unit # */
    END_OBJ* (*endLoadFunc) (char*, void*); /* The Load function. */
    char* endLoadString; /* The load string. */
    BOOL endLoan;         /* Do we loan buffers? */
    void* pBSP;           /* BSP private */
    BOOL processed;       /* Has this been processed? */
} END_TBL_ENTRY;
```

4._netDpool(_pNetDpool) _netSysPool(_pNetSysPool)

系统网络缓冲区.在netLibInit()->netLibGeneralInit()->mbinit()中初始化.

```
_netDPool(_pNetSysPool)
```

Only used for data transfer in the network stack.

```
_netsyspool(_pNetDpool)
```

Used for network stack system structures such as routes, sockets, protocol control blocks, interface addresses, mulitcast add

5.struct ifnet *ifnet; /* list of all network interfaces */

网络接口链表, 可以使用ifShow()通过遍历ifp->if_next查看整个表. 由if_attach(ifp)加入一个. ipAttach()内部调用了if_attach().

IP层自己有一个IP_DRV_CTRL ipDrvCtrl[IP_MAX_UNITS], 其中包含了struct ifnet.

最后一个成员是:

```
structifqueue if_snd; /* output queue */
```

6.UDP和TCP都有in_pcb的链表,指向每个socket的so_pcb.

```
struct inpcb
{
    LIST_ENTRY(inpcb) inp_list; /* list for all PCBs of this proto */
    LIST_ENTRY(inpcb) inp_hash; /* hash list */
    structinpcbinfo *inp_pcbinfo;
```



```
structin_addr inp_faddr; /* foreign host table entry */
u_shortinp_fport; /* foreign port */
```



```
structin_addr inp_laddr; /* local host table entry */
u_shortinp_lport; /* local port */
```



```
structsocket *inp_socket; /* back pointer to socket */
caddr_tinp_ppcb; /* pointer to per-protocol pcb */
structroute inp_route; /* placeholder for routing entry */
intinp_flags; /* generic IP/datagram flags */
```

```

structip inp_ip; /* header prototype; should have more */
structmbuf *inp_options; /* IP options */
structip_moptions *inp_moptions; /* IP multicast options */
};

```

7. struct protosw inetsw [IP_PROTO_NUM_MAX] Protocol switch table, 存放各种协议的函数。

Each protocol has a handle initializing one of these structures,

which is used for protocol-protocol and system-protocol communication.

Protocols pass data between themselves as chains of mbufs using the pr_input and pr_output hooks.

pr_input的调用

```

ipintr()
{
(*inetsw[ip_protox[ip->ip_p]].pr_input) (m, hlen);
}

```

struct protosw

```

{
    shortpr_type; /* socket type used for */
    structdomain *pr_domain; /* domain protocol a member of */
    shortpr_protocol; /* protocol number */
    shortpr_flags; /* see below */

    void(*pr_input) (); /* input to protocol (from below) */
    int(*pr_output) (); /* output to protocol (from above) */
    void(*pr_ctlinput) (); /* control input (from below) */
    int(*pr_ctloutput) (); /* control output (from above) */
    int(*pr_usrreq) (); /* user request: see list below */

    void(*pr_init) (); /* initialization hook */
    void(*pr_fasttimo) (); /* fast timeout (200ms) */
    void(*pr_slowtimo) (); /* slow timeout (500ms) */
    void(*pr_drain) (); /* flush any excess space possible */
    int(*pr_sysctl) (); /* sysctl for protocol */
};

```

在ipLibInit(), icmpLibInit(), igmpLibInit(), udpLibInit(), tcpLibInit()等都会有一个protosw.

```

{
pProtoSwitch
->pr_type= (0, SOCK_RAW , SOCK_RAW , SOCK_DGRAM, SOCK_STREAM)
->pr_domain= &inetdomain;
->pr_protocol= (IPPROTO_IP, IPPROTO_ICMP, IPPROTO_IGMP, IPPROTO_UDP, IPPROTO_TCP)
->pr_flags= PR_ATOMIC | PR_ADDR;
->pr_input= (0, icmp_input , igmp_input , udp_input, tcp_input);
->pr_output= (ip_output , rip_output, rip_output, 0, 0);
->pr_ctlinput= udp_ctlinput;
->pr_ctloutput= ip_ctloutput;
->pr_usrreq= udp_usrreq;
->pr_init= udp_init;
_protoSwIndex++;
}

```

8. struct domain *domains; /* list of domain descriptors */

由addDomain (struct domain * pDomain)添加一个新的域。

系统预定义的一个域为:

```
struct domain inetdomain =
{
    AF_INET, "internet", 0, 0, 0,
    inetsw, &inetsw[sizeof(inetsw)/sizeof(inetsw[0])], 0,
    m_inithead, 27, sizeof(struct sockaddr_in)
};
9.RIP ripState;
整个系统的rip状态.
```

10.struct radix_node_head *rt_tables[]

整个系统的路由表.可以有多种,如rt_tables[AF_INET].

三: 中断相关

11.INTR_HANDLERintrVecTable[NUM_VEC_MAX]/* Intr vector table */

MPC860 core的外部中断表, 中断定位在0x500处.由intConnect()加入中断,ppc860IntrDeMux()相应中断并分解响应.

四: 设备相关

12.DL_LIST iosDvList

所有已安装的device表, 由iosDevAdd()添加一个表项, iosDelete()删除一个表项.表项为:

```
typedef struct /* DEV_HDR - device header for all device structures */
{
    DL_NODEnode;/* device linked list node */
    shortdrvNum;/* driver number for this device,指向drvTable[]相应的位置 */
    char *name;/* device name */
} DEV_HDR;
```

13.DRV_ENTRY drvTable[NUM_DRIVERS] /* max 20 drivers in drvTable */

所有硬件驱动表.由iosDrvInstall()添加表项.其中表项为:

```
typedef struct/* DRV_ENTRY - entries in driver jump table */
{
    FUNCPTRde_create;
    FUNCPTRde_delete;
    FUNCPTRde_open;
    FUNCPTRde_close;
    FUNCPTRde_read;
    FUNCPTRde_write;
    FUNCPTRde_ioctl;
    BOOLde_inuse;
} DRV_ENTRY;
```

14.FD_ENTRY fdTable[NUM_FILES] /* max 50 files open simultaneously */

所有打开的文件描述符表.又fopen()添加一个表项,fclose()删除一个表项.其中表项为:

```
typedef struct/* FD_ENTRY - entries in file table */
{
    DEV_HDR *pDevHdr;/* device header for this file */
    int value;/* driver's id for this file */
    char *name;/* actual file name */
    BOOL inuse;/* active entry */
} FD_ENTRY;
```

五: 其他

15.

```
#define BOOT_LINE_OFFSET 0x4200
#define BOOT_LINE_ADRS((char *) (LOCAL_MEM_LOCAL_ADRS+BOOT_LINE_OFFSET))
#define DEFAULT_BOOT_LINE "ide=0,0(0,0)host:/vxWorks.dat f=0x8 tn=HC3600 o=cpm"
```

启动时,如果用户没有输入boot参数,系统便把DEFAULT_BOOT_LINE拷贝到BOOT_LINE_ADRS供 整个系统使用.

顶

0

踩

0

上一篇 字节序有大端和小端之分

下一篇 vxworks异常分析

我的同类文章

vxworks (1)		
• vxworks异常分析	2008-04-09	阅读 1103

猜你在找

数据结构基础系列(11)：文件	LINUX系统移植史上最全最细强烈推荐
数据结构基础系列(6)：树和二叉树	uboot代码详细分析pdf
数据结构基础系列(4)：串	逐行分析u-boot转
数据结构基础系列(7)：图	Bootloader之uBoot简介
CSDN攒课第二期：高并发Web网站构建和安全防护	逐行分析u-boot

查看评论

1楼 匿名用户 2010-04-10 18:08发表



发表评论

用户 名： liu1989feng

评论内容：

提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目											
全部主题	Hadoop	AWS	移动游戏	Java	Android	iOS	Swift	智能硬件	Docker	OpenStack	VPN
Spark	ERP	IE10	Eclipse	CRM	JavaScript	数据库	Ubuntu	NFC	WAP	jQuery	BI
Apache	.NET	API	HTML	SDK	IIS	Fedora	XML	LBS	Unity	Splashtop	UML
Windows Mobile	Rails	QEMU	KDE	Cassandra	CloudStack	FTC	coremail	OPhone	CouchBase	云计算	
iOS6	Rackspace	Web App	SpringSide	Maemo	Compuware	大数据	aptech	Perl	Tornado	Ruby	Hibernate
ThinkPHP	HBase	Pure	Solr	Angular	Cloud Foundry	Redis	Scala	Django	Bootstrap		