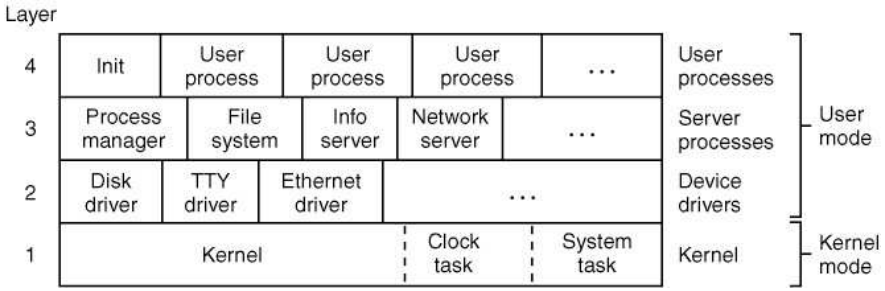
## La struttura interna di Minix 3




## La CPU e il BIOS

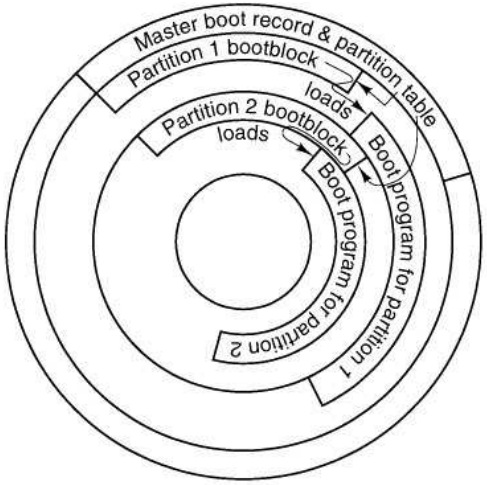
BIOS: un insieme di programmi cablati in ROM (i386-specific)

All'accensione la CPU esegue il codice all'indirizzo 0xffffff0 (mappato sul BIOS)

0. Power-On Self Test (POST)
1. Inizializza il bus PCI
2. Cerca un S.O. da caricare
3. Copia il primo settore del device di boot in RAM, e ci salta dentro

## Il boot da floppy


## Il boot da hard disk



## Il boot da hard disk

Il primo settore è il Master Boot Record (MBR)

Contiene:
- la tabella delle partizioni

Chiama:
- il programma di bootstrap bootstrap program (< 512 bytes)

bootstrap:
- carica il programma boot

boot:
- cerca la boot image

## Componenti di Minix

| **Componente** | **Descrizione** | **Caricato da** |
|---|---|---|
| kernel | kernel + clock and system tasks | boot image |
| pm | Process Manager | boot image |
| fs | File System | boot image |
| rs | Reincarnation Server | boot image |
| memory | RAM disk driver | boot image |
| log | log driver | boot image |
| tty | console and keyboard driver | boot image |
| init | genitore di tutti i processi utente | boot image |
| is | Information server (debug dumps) | /etc/rc |
| cmos | Legge l'orologio CMOS | /etc/rc |
| random | generatore di numeri casuali | /etc/rc |
| printer | printer driver | /etc/rc |

## Init

1. kernel
2. drivers (memory, log, tty)
3. servers (rs, fs, pm)
4. init(8) è il primo processo utente
   - esegue /etc/rc
     - esegue *floppy*, *is*, *cmos*
     - check disks
     - esegue i demoni *update*, *usyslogd*
   - esegue *getty* (vedi /etc/ttytab)

## La sequenza di login

init

/usr/bin/getty

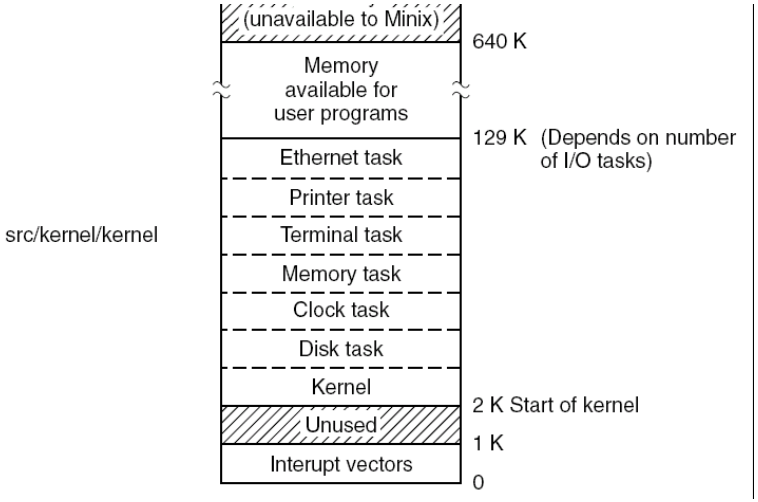
/usr/bin/login

/bin/sh

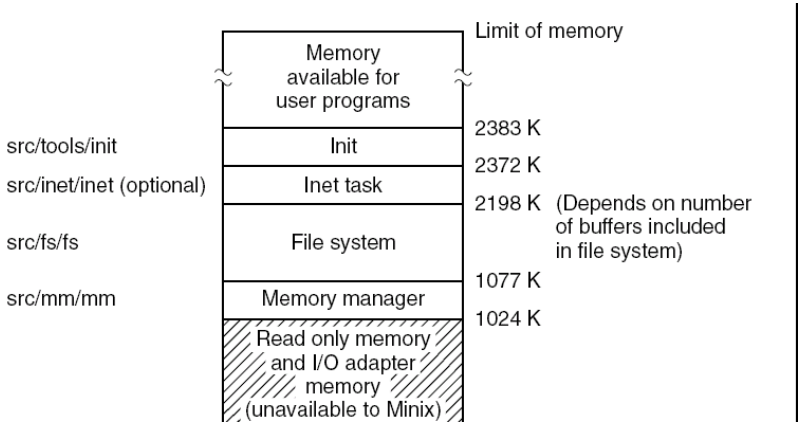
cp, ls, ... exit

(init esce dalla wait e crea un nuovo proc /usr/bin/getty)

## La memoria di Minix




## La memoria di Minix




## IPC in Minix

Tre primitive

- send(destination, &message)
- receive(source, &message)
- sendrec(src_dst, &message)

Comunicazioni ristrette

- no user-process to user-process
- si inter-layer
- si a layer sottostante

Rendezvous

## Process scheduling

Interrupt $\Rightarrow$ message

send() $\Rightarrow$ trap (software interrupt) $\Rightarrow$ message

Minix: multilevel queue scheduling

16 livelli

idle $<$ user processes $<$ servers $<$ drivers $<$ tasks

un processo che esaurisce il suo quanto viene declassato

## Organizzazione dei sorgenti

- /usr/src/
  - kernel/ (layer 1)
  - drivers/ (layer 2)
  - servers/ (layer 3)
  - commands/ (layer 4)
  - tools/ (Makefile)
  - lib/
  - boot/
- /usr/src/include/
  - sys/
  - minix/
  - ibm/
  - ...

## Kernel master include file I

src/kernel/kernel.h

```
#ifndef KERNEL_H
#define KERNEL_H

#define _POSIX_SOURCE      1    /* tell headers to include POSIX stuff */
#define _MINIX             1    /* tell headers to include MINIX stuff */
#define _SYSTEM            1    /* tell headers that this is the kernel */

/* The following are so basic, all the *.c files get them automatically. *
#include <minix/config.h>       /* global configuration, MUST be first */
#include <ansi.h>               /* C style: ANSI or K&R, MUST be second */
#include <sys/types.h>          /* general system types */
#include <minix/const.h>        /* MINIX specific constants */
#include <minix/type.h>         /* MINIX specific types, e.g. message */
#include <minix/ipc.h>          /* MINIX run-time system */
#include <timers.h>             /* watchdog timer management */
#include <errno.h>              /* return codes and error numbers */

#if (CHIP == INTEL)
#include <ibm/portio.h>         /* device I/O and toggle interrupts */
```

## Kernel master include file II

```
#endif

/* Important kernel header files. */
#include "config.h"             /* configuration, MUST be first */
#include "const.h"              /* constants, MUST be second */
#include "type.h"               /* type definitions, MUST be third */
#include "proto.h"              /* function prototypes */
#include "glo.h"                /* global variables */
#include "ipc.h"                /* IPC constants */
#include "debug.h"              /* debugging, MUST be last kernel header *

#endif /* KERNEL_H */
```

## Feature test macros: _ANSI

include/ansi.h

```
#ifndef _ANSI_H
#define _ANSI_H

#if __STDC__ == 1
#define _ANSI           31459   /* compiler claims full ANSI conformance *
#endif

#ifdef _ANSI

/* Keep everything for ANSI prototypes. */
#define _PROTOTYPE(function, params)    function params
...

#else

/* Throw away the parameters for K&R prototypes. */
#define _PROTOTYPE(function, params)    function()
...

#endif /* _ANSI */
#endif /* ANSI_H */
```

## Compatibilità K&R – ANSI

```
_PROTOTIPE(int foo, (char* a, int b, double c));
```

se vale _ANSI:

```
int foo (char* a, int b, double c);
```

altrimenti

```
int foo ();
```

## Posix definitions

include/sys/unistd.h

```
/* NULL must be defined in <unistd.h> according to POSIX Sec. 2.7.1. */
#define NULL    ((void *)0)

/* Function Prototypes. */
_PROTOTYPE( void _exit, (int _status)                                   );
_PROTOTYPE( int access, (const char *_path, int _amode)                 );
_PROTOTYPE( unsigned int alarm, (unsigned int _seconds)                 );
_PROTOTYPE( int chdir, (const char *_path)                              );
_PROTOTYPE( int chown, (const char *_path, _mnx_Uid_t _owner, _mnx_Gid_t _
_PROTOTYPE( int close, (int _fd)                                        );
_PROTOTYPE( int execl, (const char *_path, const char *_arg, ...)       );
_PROTOTYPE( int execle, (const char *_path, const char *_arg, ...)      );
_PROTOTYPE( int execlp, (const char *_file, const char *arg, ...)       );
_PROTOTYPE( int execv, (const char *_path, char *const _argv[])         );
_PROTOTYPE( int execve, (const char *_path, char *const _argv[],
                                          char *const _envp[])    );
_PROTOTYPE( int execvp, (const char *_file, char *const _argv[])        );
_PROTOTYPE( pid_t fork, (void)                                          );
```

## Errno

include/errno.h

```
#ifndef _ERRNO_H                /* check if <errno.h> is already included
#define _ERRNO_H                /* it is not included; note that fact */

/* Now define _SIGN as "" or "-" depending on _SYSTEM. */
#ifdef _SYSTEM
#   define _SIGN         -
#   define OK            0
#else
#   define _SIGN
#endif

extern int errno;                /* place where the error numbers go */

/* Here are the numerical values of the error numbers. */
#define _NERROR              70  /* number of errors */

#define EGENERIC      (_SIGN 99)  /* generic error */
#define EPERM         (_SIGN  1)  /* operation not permitted */
#define ENOENT        (_SIGN  2)  /* no such file or directory */
...
#endif /* _ERRNO_H */
```

## Type definitions

include/sys/types.h

```
typedef unsigned int size_t;
typedef long time_t;                /* time in sec since 1 Jan 1970 0000 GM

typedef unsigned char   u8_t;       /* 8 bit type */
typedef unsigned short u16_t;       /* 16 bit type */
typedef unsigned long  u32_t;       /* 32 bit type */

typedef char            i8_t;       /* 8 bit signed type */
typedef short          i16_t;       /* 16 bit signed type */
typedef long           i32_t;       /* 32 bit signed type */

/* Types used in disk, inode, etc. data structures. */
typedef short          dev_t;       /* holds (major|minor) device pair */
typedef char           gid_t;       /* group id */
typedef unsigned long  ino_t;       /* i-node number (V3 filesystem) */
typedef unsigned short mode_t;      /* file type and permissions bits */
typedef short        nlink_t;       /* number of links to a file */
typedef unsigned long  off_t;       /* offset within a file */
typedef int            pid_t;       /* process id (must be signed) */
typedef short          uid_t;       /* user id */
typedef unsigned long zone_t;       /* zone number */
```

## File di configurazione: minix/sys_config.h I

include/minix/sys_config.h

```
#define _MINIX_MACHINE       _MACHINE_IBM_PC

#define _MACHINE_IBM_PC            1   /* any  8088 or 80x86-based system
#define _MACHINE_ATARI            60   /* ATARI ST/STe/TT (68000/68030) *
...
#define _NR_PROCS       100
#define _NR_SYS_PROCS   32
...
#define _CHIP_INTEL             1       /* CHIP type for PC, XT, AT, 386 a
#define _CHIP_M68000            2       /* CHIP type for Atari, Amiga, Mac

#if (_MINIX_MACHINE == _MACHINE_IBM_PC)
#define _MINIX_CHIP          _CHIP_INTEL
#endif

#if (_MINIX_MACHINE == _MACHINE_ATARI) || (_MINIX_MACHINE == _MACHINE_MACI
#define _MINIX_CHIP         _CHIP_M68000
#endif
```

## File di configurazione: minix/sys_config.h II

```
#ifndef _MINIX_MACHINE
error ''In <minix/sys_config.h> please define _MINIX_MACHINE''
#endif

#ifndef _MINIX_CHIP
error ''In <minix/sys_config.h> please define _MINIX_MACHINE to have a lega
#endif

#if (_MINIX_MACHINE == 0)
error ''_MINIX_MACHINE has incorrect value (0)''
#endif
```

## File di configurazione: src/minix/config.h I

```
#define OS_RELEASE ''3''
#define OS_VERSION ''1.1''

#include <minix/sys_config.h>

#if _MINIX_SMALL

#define NR_BUFS 100

#else

/* The buffer cache should be made as large as you can afford. */
#if (MACHINE == IBM_PC && _WORD_SIZE == 2)
#define NR_BUFS           40    /* # blocks in the buffer cache */
#endif

#if (MACHINE == IBM_PC && _WORD_SIZE == 4)
#define NR_BUFS         1200    /* # blocks in the buffer cache */
#endif
```

## File di configurazione: src/minix/config.h II

```
#endif  /* _MINIX_SMALL */

#define NR_CONS             4    /* # system consoles (1 to 8) */
#define NR_RS_LINES         4    /* # rs232 terminals (0 to 4) */
#define NR_PTYS            32    /* # pseudo terminals (0 to 64) */

/*===========================================================================
 *      There are no user-settable parameters after this line
 *===========================================================================
```

## Costanti: include/minix/const.h

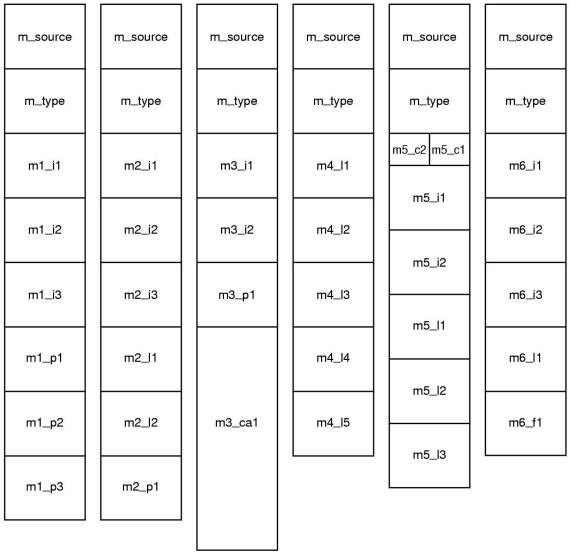
```
#define EXTERN        extern    /* used in *.h files */
#define PRIVATE       static    /* PRIVATE x limits the scope of x */
#define PUBLIC                  /* PUBLIC is the opposite of PRIVATE */
#define FORWARD       static    /* some compilers require this to be 'stat

#define TRUE               1    /* used for turning integers into Booleans
#define FALSE              0    /* used for turning integers into Booleans

#define HZ                60    /* clock freq (software settable on IBM-PC

#define SUPER_USER (uid_t) 0    /* uid_t of superuser */
```

## Minix message types




## Inter-process communication: include/minix/ipc.h I

```
typedef struct int m1i1, m1i2, m1i3; char *m1p1, *m1p2, *m1p3; mess_1;
typedef struct int m2i1, m2i2, m2i3; long m2l1, m2l2; char *m2p1; mess_2;
typedef struct int m3i1, m3i2; char *m3p1; char m3ca1[M3_STRING]; mess_3;
...

typedef struct {
  int m_source;                 /* who sent the message */
  int m_type;                   /* what kind of message is it */
  union {
        mess_1 m_m1;
        mess_2 m_m2;
        mess_3 m_m3;
        mess_4 m_m4;
        mess_5 m_m5;
        mess_7 m_m7;
        mess_8 m_m8;
  } m_u;
} message;
```

## Inter-process communication: include/minix/ipc.h II

```
/* The following defines provide names for useful members. */
#define m1_i1  m_u.m_m1.m1i1
#define m1_i2  m_u.m_m1.m1i2
...

_PROTOTYPE( int notify, (int dest)                                  );
_PROTOTYPE( int sendrec, (int src_dest, message *m_ptr)             );
_PROTOTYPE( int receive, (int src, message *m_ptr)                  );
_PROTOTYPE( int send, (int dest, message *m_ptr)                    );
```

## I numeri dei messaggi (e system calls)

include/minix/callnr.h

```
#define NCALLS            91    /* number of system calls allowed */

#define EXIT               1
#define FORK               2
#define READ               3
#define WRITE              4
#define OPEN               5
#define CLOSE              6
#define WAIT               7
#define CREAT              8
#define LINK               9
#define UNLINK            10
#define WAITPID           11
#define CHDIR             12
#define TIME              13
#define MKNOD             14
#define CHMOD             15
#define CHOWN             16
#define BRK               17
```

## Costanti comuni per ipc: include/minix/com.h I

```
/*===========================================================================*
 *                               Magic process numbers                       *
 *===========================================================================*/

#define ANY             0x7ace  /* used to indicate 'any process' */
#define NONE            0x6ace  /* used to indicate 'no process at all' */
#define SELF            0x8ace  /* used to indicate 'own process' */

/*===========================================================================*
 *              Process numbers of processes in the system image             *
 *===========================================================================*/

/* Kernel tasks. These all run in the same address space. */
#define IDLE              -4    /* runs when no one else can run */
#define CLOCK             -3    /* alarms and other clock functions */
#define SYSTEM            -2    /* request system functionality */
#define KERNEL            -1    /* pseudo-process for IPC and scheduling */
#define HARDWARE     KERNEL     /* for hardware interrupt handlers */
```

## Costanti comuni per ipc: include/minix/com.h II

```
/* Number of tasks. Note that NR_PROCS is defined in <minix/config.h>. */
#define NR_TASKS          4

/* User-space processes, that is, device drivers, servers, and INIT. */
#define PM_PROC_NR        0     /* process manager */
#define FS_PROC_NR        1     /* file system */
#define RS_PROC_NR        2     /* reincarnation server */
#define MEM_PROC_NR       3     /* memory driver (RAM disk, null, etc.) */
#define LOG_PROC_NR       4     /* log device driver */
#define TTY_PROC_NR       5     /* terminal (TTY) driver */
#define DRVR_PROC_NR      6     /* device driver for boot medium */
#define DS_PROC_NR        7     /* data store server */
#define INIT_PROC_NR      8     /* init - goes multiuser */

/* Number of processes contained in the system image. */
#define NR_BOOT_PROCS    (NR_TASKS + INIT_PROC_NR + 1)
```

## Types again: kernel/type.h I

```
#if (CHIP == INTEL)
typedef unsigned reg_t;         /* machine register */

/* The stack frame layout is determined by the software, but for efficienc
 * it is laid out so the assembly code to use it is as simple as possible.
 */
struct stackframe_s {           /* proc_ptr points here */
  u16_t gs;                     /* last item pushed by save */
  u16_t fs;                     /*  ^ */
  u16_t es;                     /*  | */
  u16_t ds;                     /*  | */
  reg_t di;                     /* di through cx are not accessed in C */
  reg_t si;                     /* order is to match pusha/popa */
  reg_t fp;                     /* bp */
  reg_t st;                     /* hole for another copy of sp */
  reg_t bx;                     /*  | */
  reg_t dx;                     /*  | */
  reg_t cx;                     /*  | */
  reg_t retreg;                 /* ax and above are all pushed by save */
```

## Types again: kernel/type.h II

```
  reg_t retadr;                 /* return address for assembly code save()
  reg_t pc;                     /*  ^  last item pushed by interrupt */
  reg_t cs;                     /*  |  */
  reg_t psw;                    /*  |  */
  reg_t sp;                     /*  |  */
  reg_t ss;                     /* these are pushed by CPU during interrup
};
```

## Kernel global variables: kernel/glo.h I

```
#ifdef _TABLE
#undef EXTERN
#define EXTERN
#endif

#include <minix/config.h>
#include config.h

/* Kernel information structures. This groups vital kernel information. */
EXTERN phys_bytes aout;                 /* address of a.out headers */
EXTERN struct kinfo kinfo;              /* kernel information for users */
EXTERN struct machine machine;          /* machine information for users *
EXTERN struct kmessages kmess;          /* diagnostic messages in kernel *
EXTERN struct randomness krandom;       /* gather kernel random informatio

/* Process scheduling information and the kernel reentry count. */
EXTERN struct proc *prev_ptr;   /* previously running process */
EXTERN struct proc *proc_ptr;   /* pointer to currently running process */
EXTERN struct proc *next_ptr;   /* next process to run after restart() */
```

## Kernel global variables: kernel/glo.h II

```
EXTERN struct proc *bill_ptr;   /* process to bill for clock ticks */
EXTERN char k_reenter;          /* kernel reentry count (entry count less
EXTERN unsigned lost_ticks;     /* clock ticks counted outside clock task
```

## La tabella dei processi: kernel/proc.h I

```
struct proc {
  struct stackframe_s p_reg;    /* process' registers saved in stack frame

  proc_nr_t p_nr;               /* number of this process (for fast access
  struct priv *p_priv;          /* system privileges structure */
  char p_rts_flags;             /* SENDING, RECEIVING, etc. */

  char p_misc_flags;            /* Flags that do suspend the process */

  char p_priority;              /* current scheduling priority */
  char p_max_priority;          /* maximum scheduling priority */
  char p_ticks_left;            /* number of scheduling ticks left */
  char p_quantum_size;          /* quantum size in ticks */

  struct mem_map p_memmap[NR_LOCAL_SEGS];   /* memory map (T, D, S) */

  clock_t p_user_time;          /* user time in ticks */
  clock_t p_sys_time;           /* sys time in ticks */
```

## La tabella dei processi: kernel/proc.h II

```
  struct proc *p_nextready;     /* pointer to next ready process */
  struct proc *p_caller_q;      /* head of list of procs wishing to send *
  struct proc *p_q_link;        /* link to next proc wishing to send */
  message *p_messbuf;           /* pointer to passed message buffer */
  proc_nr_t p_getfrom;          /* from whom does process want to receive?
  proc_nr_t p_sendto;           /* to whom does process want to send? */

  sigset_t p_pending;           /* bit map for pending kernel signals */

  char p_name[P_NAME_LEN];      /* name of the process, including \0 */
};

/* Bits for the runtime flags. A process is runnable iff p_rts_flags == 0.
#define SLOT_FREE       0x01    /* process slot is free */
#define NO_MAP          0x02    /* keeps unmapped forked child from runnin
#define SENDING         0x04    /* process blocked trying to SEND */
#define RECEIVING       0x08    /* process blocked trying to RECEIVE */
#define SIGNALED        0x10    /* set when new kernel signal arrives */
#define SIG_PENDING     0x20    /* unready while signal being processed */
#define P_STOP          0x40    /* set when process is being traced */
#define NO_PRIV         0x80    /* keep forked system process from running
```

## La tabella dei processi: kernel/proc.h III

```
/* Scheduling priorities for p_priority. Values must start at zero (highes
 * priority) and increment.  Priorities of the processes in the boot image
 * can be set in table.c. IDLE must have a queue for itself, to prevent lo
 * priority user processes to run round-robin with IDLE.
 */
#define NR_SCHED_QUEUES   16    /* MUST equal minimum priority + 1 */
#define TASK_Q             0    /* highest, used for kernel tasks */
#define MAX_USER_Q         0    /* highest priority for user processes */
#define USER_Q             7    /* default (should correspond to nice 0) *
#define MIN_USER_Q        14    /* minimum priority for user processes */
#define IDLE_Q            15    /* lowest, only IDLE process goes here */

/* Magic process table addresses. */
#define BEG_PROC_ADDR (&proc[0])
#define BEG_USER_ADDR (&proc[NR_TASKS])
#define END_PROC_ADDR (&proc[NR_TASKS + NR_PROCS])

#define NIL_PROC          ((struct proc *) 0)
#define NIL_SYS_PROC      ((struct proc *) 1)
#define cproc_addr(n)     (&(proc + NR_TASKS)[(n)])
```

## La tabella dei processi: kernel/proc.h IV

```
#define proc_addr(n)      (pproc_addr + NR_TASKS)[(n)]
#define proc_nr(p)        ((p)->p_nr)

#define isokprocn(n)      ((unsigned) ((n) + NR_TASKS) < NR_PROCS + NR_TAS
#define isemptyn(n)       isemptyp(proc_addr(n))
#define isemptyp(p)       ((p)->p_rts_flags == SLOT_FREE)
#define iskernelp(p)      iskerneln((p)->p_nr)
#define iskerneln(n)      ((n) < 0)
#define isuserp(p)        isusern((p)->p_nr)
#define isusern(n)        ((n) >= 0)

/* The process table and pointers to process table slots. The pointers all
 * faster access because now a process entry can be found by indexing the
 * pproc_addr array, while accessing an element i requires a multiplicatio
 * with sizeof(struct proc) to determine the address.
 */
EXTERN struct proc proc[NR_TASKS + NR_PROCS];   /* process table */
EXTERN struct proc *pproc_addr[NR_TASKS + NR_PROCS];
EXTERN struct proc *rdy_head[NR_SCHED_QUEUES]; /* ptrs to ready list heade
EXTERN struct proc *rdy_tail[NR_SCHED_QUEUES]; /* ptrs to ready list tails
```