

```

1   Wichtige NR's
2   NR_REGS
3   NR_SEGS
4   NR_TASKS
5   NR_PROCS
6   NR_PROCS
7   NR_HOLES

8   Wichtige Strukturen

9   struct mem_map {
10      vir_clicks mem_vir;
11      phys_clicks mem_phys;
12      vir_clicks mem_len;
13 };

14 EXTERN struct proc {
15      int p_reg [NR_REGS];
16      ...
17      struct mem_map [NR_SEGS];
18      ...
19 } proc [NR_TASKS + NR_PROCS];

20 PRIVATE struct hole {
21      phys_clicks h_base;
22      phys_clicks h_len;
23      struct hole *h_next;
24 } hole [NR_HOLES];

25      typedef      unsigned      vir_bytes;
26      typedef      unsigned      vir_clicks;
27      typedef      long           phys_bytes;
28      typedef      unsigned      phys_clicks;
29      typedef      int            signed_clicks;

30   Wichtig

31   1. Kernel
32   2. Memory Manager
33   3. File System, Dateisystem

34   kernel
35   mm
36   fs

37   h
38   kernel
39   mm
40   fs

```

```

41 h
42     h/const.h
43     h/callnr.h
44     h/com.h
45     h/error.h
46     h/sgtty.h
47     h/signal.h
48     h/stat.h
49     h/type.h
50 kernel
51     kernel/const.h
52     kernel/glo.h
53     kernel/proc.h
54     kernel/type.h
55     kernel/main.c
56     kernel/mpx88.s
57     kernel/klib88.s
58     kernel/proc.c
59     kernel/memory.c
60     kernel/floppy.c
61     kernel/clock.c
62     kernel/tty.c
63     kernel/system.c
64 mm
65     mm/const.h
66     mm/glo.h
67     mm/mproc.h
68     mm/param.h
69     mm/type.h
70     mm/main.c
71     mm/forkexit.c
72     mm/exec.c
73     mm/break.c
74     mm/signal.c
75     mm/getset.c
76     mm/alloc.c
77
78 fs
79 h
80     h/const.h
81     h/callnr.h
82     h/com.h
83     h/error.h
84     h/sgtty.h
85         Datenstrukturen fuer IOCTL
86             struct sgtyb {...}
87             struct tchars {...}
88     h/signal.h
89         NR_SIGS

```

```

90     N_SIGS
91     SIGHUP
92     SIGINT
93     SIGQUIT
94     SIGILL
95     SIGTRAP
96     SIGIOT
97     SIGEMT
98     ...
99     SIGKILL
100    ...
101    SIGPIPE
102    SIGSYS
103    SIGALARM
104    SIGTERM
105    h/stat.h
106    h/type.h
107        typedef    unsigned    vir_bytes;
108        typedef    unsigned    vir_clicks;
109        typedef    long    phys_bytes;
110        typedef    unsigned    phys_clicks;
111        typedef    int    signed_clicks;
112    kernel
113        kernel/const.h
114            NR_REGS
115            ES_REG
116            DS_REG
117            CS_REG
118            SS_REG
119            Interrupt Vektoren: DIVIDE_VECTOR, CLOCK_VECTOR, KEYBOARD_VECTOR, FLOPPY_VECTOR
120            PIC 8259A
121        kernel/glo.h
122        kernel/proc.h
123        kernel/type.h
124        kernel/main.c
125            main ()
126            unexpected_int ()
127            trap ()
128            div_trap ()
129            panic ()
130            set_vec (vec_nr, addr, base_click)
131        kernel/mpx88.s
132            MINIX
133            s_call
134            tty_int
135            lpr_int
136            disk_int
137            wini_int
138            clock_int
139            suprise

```

```
140         trp
141         divide
142         save
143         restart
144         idle
145         data
146     kernel/klib88.s
147         phys_copy
148         cp_mess
149         port_out
150         port_in
151         lock
152         unlock
153         restore
154         build_sig
155         csv
156         cret
157         get_chrome
158         dma_read
159         vid_copy
160         get_byte
161         reboot
162         wreboot
163     kernel/proc.c
164         sys_call
165         interrupt
166         ready
167         unready
168         sched
169         mini_send
170         pick_proc
171     kernel/memory.c
172         mem_task
173         do_mem
174         do_setup
175     kernel/floopy.c
176         floppy_task
177         do_rdw
178         dma_setup
179         start_motor
180         stop_motor
181         seek
182         transfer
183         fdc_results
184         fdc_out
185         recalibrate
186         reset
187         clock_mess
188         send_mess
189     kernel/clock.c
```

```

190         clock_task
191         do_setalarm
192         do_get_time
193         do_set_time
194         do_clock_tick
195         accounting
196         init_clock
197     kernel/tty.c
198         tty_task // Hauptroutine
199         do_charint
200         in_char
201         make_break
202         echo
203         chuck
204         do_read
205         rd_chars
206         finish
207         do_write
208         do_ioctl
209         do_cancel
210         tty_reply
211         sigchar
212         keyboard
213         console
214         out_char
215         scroll_screen
216         flush
217         move_to
218         escape
219         set_6845
220         beep
221         tty_init
222         putc
223         func_key
224     kernel/system.c
225         sys_task
226         do_fork
227         do_newmap
228         do_exec
229         do_xit
230         do_getsp
231         do_times
232         do_abort
233         do_sig
234         do_copy
235         cause_sig
236         inform
237         umap
238     mm
239         mm/const.h

```

```
240 mm/glo.h
241 mm/mproc.h
242 mm/param.h
243 mm/type.h
244 mm/main.c
245     main
246     get_work
247     reply
248     mm_init
249     do_brk2
250     set_map
251 mm/forkexit.c
252     do_fork
253     do_mm_exit
254     mm_exit
255     do_wait
256     cleanup
257 mm/exec.c
258     do_exec
259     read_header
260     new_mem
261     patch_ptr
262     load_seg
263 mm/break.c
264     do_brk
265     adjust
266     size_ok
267     stack_fault
268 mm/signal.c
269     do_signal
270     do_kill
271     do_ksig
272     check_sig
273     sig_proc
274     do_alarm
275     set_alarm
276     do_pause
277     unpause
278     dump_core
279 mm/getset.c
280     do_getset
281 mm/alloc.c
282     alloc_mem
283     free_mem
284     del_slot
285     merge
286     max_hole
287     mem_init
288 fs
```

```

289 1 h/const.h
290 h/const.h
291
292 #define      EXTERN          extern
293 #define      PRIVATE        static
294 #define      PUBLIC
295 #define      FORWARD
296 #define      TRUE           1
297 #define      FALSE          0
298 ...
299 #define      BLOCK_SIZE     1024
300 #define      SUPER_USER     (uid) 0
301 #define      MAJOR           8
302 #define      MINOR           0
303 #define      NR_TASKS       8
304 #define      NR_PROCS       16
305 #define      NR_SEGS        3
306 #define      T               0
307 #define      D               1
308 #define      S               2
309
310     Wichtig
311
312     1. Text
313
314     2. Data
315
316     3. Stack
317
318 Wichtig! Code = Text
319 Wichtig:
320
321     1. Text T
322
323     2. Data D
324
325     3. Stack S
326
327 Wichtig:
328
329     1. Text T, 0
330
331     2. Data D, 1
332
333     3. Stack S, 2
334
335 Wichtig:
336
337     1. NR_TASKS
338
339     2. NR_PROCS
340
341     3. NR_SEGS

```

326     So, achtung aufgepasst!

```

327 struct proc {
328
329
330 } proc [NR_TASKS + NR_PROCS]

```

- 331     1. Anzahl der Prozesse
- 332     2. Anzahl der Tasks
- 333     1. Im Speichermanager
  - 334         (a) NR\_PROCS - Im Speichermanager, nur Prozesse
- 335     2. Im Kernel
  - 336         (a) NR\_TASKS - im Kernel, Tasks
  - 337         (b) NR\_PROCS - Kernel, Prozesse

338     Dinge mit NR

- 339     1. NR\_TASKS
- 340     2. NR\_PROCS
- 341     3. NR\_SEGS

342     Dinge mit NR zur Struktur

- 343     1. (NR\_TASKS, proc)
- 344     2. (NR\_PROCS, proc)
- 345     3. (NR\_SEGS, mem\_map)

346     Wichtig

- 347     1. mem - speicher
- 348     2. map - karte

349     Wichtig

- 350     1. NR\_SEGS
- 351     2. T
- 352     3. D
- 353     4. S

354     Wichtig

- 355     1. NR\_SEGS, 3, maximal grenze
- 356     2. T, 0, erstes Element
- 357     3. D, 1, zweites Element
- 358     4. S, 2, drittes Element



## 359 2 h/callnr.h

360 Wichtig

361 h/callnr.h ^= die wichtigsten Systemaufrufe

362 Alle lernen

```
363 #define          NCALLS          69
364 #define          EXIT            1
365 #define          FORK            2
366 #define          READ            3
367 #define          WRITE           4
368 #define          OPEN            5
369 #define          CLOSE           6
370 #define          WAIT            7
371 #define          CREAT           8
372 ...
```

373 Die wichtigsten Systemaufrufe. Wichtig

374 NCALLS ^= Anzahl der Systemaufrufe

375 1. exit ()

376 2. fork ()

377 3. read ()

378 4. write ()

379 5. open ()

380 6. close ()

381 7. create ()

382 Daneben gibt es, einfach lernen, egal, wie

```
383 exit
384 fork
385 read
386 write
387 open
388 close
389 wait
390 create
391 link
392 unlink
393 chdir
394 time
395 mknod
396 chmod
397 chown
```

```
398 brk
399 stat
400 lseek
401 getpid
402 mount
403 umount
404 setuid
405 getuid
406 stime
407 alarm
408 fstat
409 pause
410 utime
411 access
412 sync
413 kill
414 dup
415 pipe
416 times
417 setgid
418 getgid
419 signal
420 ioctl
421 exec
422 umask
423 chroot
```

### 424 **3 h/com.h**

425 Wichtig, Definitionen für Systemaufrufe  
426 Botschaften

```
427 SEND          1  /* Code fuer Botschaften verschicken */
428 RECEIVE       2  /* Code fuer Botschaften empfangen */
429 BOTH          3  /* Code fuer SEND + RECIEVE */
430 ANY
```

431 Wichtig  
432 Botschaften

```
433 HARDWARE
434 SYSTASK
435 CLOCK
436 MEM
437 FLOPPY
438 WINCHESTER
439 TTY
440 PRINTER
```

441 Wichtig: Botschaften, darin gibt es entsprechende Parameter - Achtung,  
442 koennte man meinen. Das sind Botschaften,

```

443 1. Fuer jedes Gerat eine Botschaft
444     (a) Hardware
445     (b) Systemaufrufe
446     (c) Uhr
447     (d) Arbeitsspeicher, MEM
448     (e) Floppy Disk
449     (f) Festplatte
450     (g) TTY
451     (h) Drucker
452
453 2. fuer alle diese gibt es Botschaften
454
455 3. Das sind keine Paramter, sondern Botschaften an das Gerat
456
457 4. Bei der Festplatte gibt es: DISK_READ, DISK_WRITE. Botschaft an das
458     Gerat: Lesen, schreiben
459
460 HARDWARE
461 SYSTASK
462     /* zum Beispiel */
463     SYS_FORK
464     SYS_EXEC
465     SYS_COPY
466     SYS_TIMES
467 CLOCK
468     SET_ALARM
469     CLOCK_TIME
470     GET_TIME
471     SET_TIME
472     REAL_TIME
473 MEM
474     RAM_DEV           /dev/ram
475     MEM_DEV           /dev/mem
476     KMEM_DEV          /dev/kmem
477     NULL_DEV          /dev/null
478 FLOPPY
479 WINCHESTER
480     DISKINT
481     DISK_READ
482     DISK_WRITE
483     DISK_IOCTL
484 TTY/PRINTER
485     . . .
486     TTY_READ
487     TTY_WRITE
488     TTY_IOCTL

```

## 485 **4 h/error.h**

486 Lauter Fehler

487 **ERROR**

488 **OK**

489 **..**

490 **EIO**

491 **ENOMEM**

492 **ENODEV**

493 **...**

494

## 495 **5 h/type.h**

```
496         typedef      unsigned      vir_bytes;
497         typedef      unsigned      vir_clicks;
498         typedef      long           phys_bytes;
499         typedef      unsigned      phys_clicks;
500         typedef      int            signed_clicks;
```

501 Lerne zu unterscheiden

502 1. Bytes

503 2. Clicks

504 Lerne zu unterscheiden

505 1. Virtuell

506 2. Physikalisch

507 1. Virtuelle Bytes

508 2. Virtuelle Clicks

509 3. Physikalische Bytes

510 4. Physikalische Clicks

```
511 struct mem_map {
512     vir_clicks mem_vir;
513     phys_clicks mem_phys;
514     vir_clicks mem_len;
515 };
```

516 1. Anfang

517 2. Länge

518 Lerne zu unterscheiden

519 1. Anfang

- 520 2. Ende
- 521 3. Laenge
- 522 Wichtig:
- 523 1. Anfang
- 524 2. Länge
- 525 wichtig:
- 526 1. Angabe in Clicks nicht in Bytes
- 527 2. Anfang:
  - 528 (a) Virtuell
  - 529 (b) Physikalisch
- 530 3. Laenge: Virtuell

## 531 **6 kernel/const.h**

- 532 1. Register
- 533 2. Interrupt-Vektoren
- 534 3. 8259A
- 535 1. Register
  - 536 (a) Übersicht
    - 537 i. **ax** 1
    - 538 ii. **bx** 2
    - 539 iii. **dx** 3
    - 540 iv. **cx** 4
    - 541 v. **si** 5
    - 542 vi. **di** 5
    - 543 vii. **sp** 7
    - 544 viii. **bp** 8
    - 545 ix. **es**
    - 546 x. **ss** 9
    - 547 xi. **ds** 10
    - 548 xii. **cs** 11
    - 549 xiii. **ip**
  - 550 (b) Definitionen
    - 551 i. Anzahl
    - 552 ii. Code
    - 553 iii. Daten
    - 554 iv. Stack

```

555     2. Interrupt-Vektoren
556         (a) Geteilt durch 0
557         (b) Uhr
558         (c) Keyboard
559         (d) XT-Wini
560         (e) Floppy
561         (f) Printer
562         (g) Systemaufrufe
563         (h) AT-Wini
564     3. 8259A
565         (a) IO-Port: 0x20, I/O-Port Interrupt Controller 0xA0, I/O-Port zweiter
566             Interrupt Controller
567         (b) Maske, 1. und 2.

568 #define     NR_REGS             11
569 #define     ES_REG              7
570 #define     DS_REG              8
571 #define     CS_REG              9
572 #define     SS_REG             10

573 Interrupt Vektoren
574 #define     DIVIDE_VECTOR       0
575 #define     CLOCK_VECTOR        8
576 #define     KEYBOARD_VECTOR     9
577 #define     XT_WINI_VECTOR     13
578 #define     FLOPPY_VECTOR      14
579 #define     PRINTER_VECTOR     15
580 #define     SYS_VECTOR         32
581 #define     ATA_WINI_VECTOR    118

582 8259A
583 #define     INT_CTL             0x20      /* I/O-Port fuer Interrupt Controller */
584 #define     INT_CTLMASK        0x21      /* Bits auf diesem Port setzen verbietet Inter
585 #define     INT2_CTL           0xA0
586 #define     INT2_MASK          0xA1
587     ...

588 7 kernel/glo.h

589 8 kernel/proc.h
590 EXTERN struct proc {
591
592 } proc [NR_TASKS + NR_PROCS];

```

```

593
594 NR_TASKS!
595 NR_PROCS!
596
597 struct proc !
598
598 EXTERN struct proc {
599     int p...
600 } proc [NR_TASKS + NR_PROCS];
601
601 EXTERN struct proc {
602     int p...1
603     int p...2
604     int p...3
605 } proc [NR_TASKS + NR_PROCS];
606
606 EXTERN struct proc {
607     int p_reg [NR_REGS];
608     ...
609     struct mem_map [NR_SEGS];
610     ...
611 } proc [NR_TASKS + NR_PROCS];
612
613 NR_REGS !
614 NR_TASKS !
615 NR_PROCS !
616 NR_SEGS !
617
617     1. NR_REGS
618
618     2. NR_TASKS
619
619     3. NR_PROCS
620
620     4. NR_SEGS
621
621     OK
622
622 EXTERN struct proc {
623     int p_reg [NR_REGS];
624     ...
625     struct mem_map [NR_SEGS];
626     int p_pid;
627     ...
628 } proc [NR_TASKS + NR_PROCS];
629
629     1. Register
630
630     2. Speicherabbild
631
631     3. Prozess ID

```

```

632 EXTERN struct proc {
633     int p_reg [NR_REGS];
634     int *p_sp;                               /* Stack pointer */
635     struct pc_psw p_pcpsw;                   /* pc und psw bei Interrupt gesichert */
636     int p_flags;                             /* P_SLOT, ... nicht flags */
637     struct mem_map [NR_SEGS];
638     int *p_splimit;                          /* Niedrigster wert fuer den Stack */
639     int p_pid;
640
641     real_time user_time;
642     real_time sys_time;
643     real_time child_utime;
644     real_time child_stime;
645     real_time p_alarm;
646
647     struct proc *p_callreq;                  /* Anfang der Liste aller Prozesse die senden woll
648     struct proc *p_sendlink;
649     message *p_messbuf;                    /* Zeiger auf Puffer fuer die Botschaft */
650     int p_getfrom;
651
652     struct proc *p_nextready;
653     int p_pending;
654 } proc [NR_TASKS + NR_PROCS];

```

## 655 9 kernel/type.h

```

656 struct pc_psw {
657     int (*pc)();
658     phys_clicks cs;
659     unsigned psw;                          /* Programm status word */
660 };

```

## 661 10 kernel/main.c

```

662 main ()
663 unexpected_int ()
664 trap ()
665 div_trap ()
666 panic ()
667 set_vec (vec_nr, addr, base_click)

```

## 668 11 kernel/mpx88.s

```

669     kernel/mpx88.s
670     MINIX
671     s_call
672     tty_int
673     lpr_int

```



```

674         disk_int
675         wini_int
676         clock_int
677         suprise
678         trp
679         divide
680         save
681         restart
682         idle
683         data

684     1. TTY
685     2. Disk : Floppy-Disk
686     3. Winchester: Hard Drive
687     4. Clock
688     5. LPR - Drucker
689
690
691     6. Systemaufrufe
692
693
694     7. idle : nichts zu tun

695 Beispiele:
696 _tty_int:
697     call save
698     call _keyboard
699     jmp _restart
700 _lpr_int:
701     call save
702     call _pr_char
703     jmp _restart
704 ...
705
706 _disk_int:
707     call save
708     mov _int_mess+2, *DISKINT
709     mov ax, #int_mess
710     push ax
711     mov ax, *FLOPPY
712     push ax
713     call _interrupt
714     jmp _start
715
716 _wini_int:
717     ...

```

```
718     mov ax, *WINI
719     ...
```

- 720 1. Unterste Ebene der Minix-kernels
- 721 2. Prozess und Botschaftsverwaltung wird abgewickelt
- 722 3. dazu in `proc.c`
- 723
- 724
- 725 4. Prozess
- 726 5. Botschaft

## 727 **12 kernel/klib88.s**

728 Wichtig:

- 729 1. Assembler-Hilfsroutinen
- 730 2. Zum Beispiel:

- 731 (a) `port_in`
- 732 (b) `port_out`

```
733 kernel/klib88.s
734     phys_copy    // kopiert Daten innerhalb des Speichers
735     cp_mess     // kopiert Botschaften von der Quelle zum ziel
736     port_out
737     port_in
738     lock        // interrupts verbieten
739     unlock     // interrupts zulassen
740     restore
741     build_sig
742     csv
743     cret
744     get_chrome  // liefert 0 bei Monochrom Bildschirm, 1 bei Farbbildschirm
745     dma_read
746     vid_copy   // Diese Routine nimmt einen String von Zeichen und schreibt ihn a
747     get_byte   // Holt Byte irgendwo aus dem Speicher
748     reboot
749     wreboot
```

## 750 **13 kernel/proc.c**

- 751 1. Prozessabarbeitung
- 752 2. Botschaftsabarbeitung

753 Wichtig:

- 754 1. ready: Prozess in Warteschlange  
755 2. unready: Prozess aus Warteschlange entfernen  
756 3. sched: Prozess zu lange gerechnet

757 sys\_call  
758 interrupt  
759 ready  
760 unready  
761 sched  
762 mini\_send  
763 pick\_proc

## 764 14 kernel/memory.c

765 Treiber fuer vier wichtige Geraedateien  
766 /dev/null - null Device = Daten Senke  
767 /dev/mem - ganzer Speicher  
768 /dev/kmem - vom kernel belegter Speicher  
769 /dev/ram - RAM-Disk

- 770 1. Daten Senke /dev/null  
771 2. Ganzer Speicher /dev/mem  
772 3. Speicher, Kernel /dev/kmem  
773 4. RAM-Disk /dev/ram  
774  
775 5. Treiber  
776 6. Treiber, fuer alles, was im RAM als Gerät angesprochen wird, wie bei  
777 Floppy

778 mem\_task  
779 do\_mem  
780 do\_setup

## 781 15 kernel/floppy.c

- 782 1. Treiber für Floppy-Disk-Controller  
783 2. NEC PD765-Chip

784 IO-Ports:  
785 DOR  
786 FDC\_STATUS  
787 FDC\_DATA  
788 FDC\_RATE  
789 DMA\_ADDR

790  
791 **Statusregister:**  
792 **ST0**  
793 **ST1**  
794 **ST2**  
795 **ST\_CYL**  
796 **ST\_HEAD**  
797 **ST\_SEC**  
798 **ST\_PCN**

799  
800  
801 **Kommandobytes:**  
802 **FDC\_SEEK**  
803 **FDC\_READ**  
804 **FDC\_WRITE**  
805 **FDC\_RECALLIBRATE**

806  
807 **Parameter**  
808 **SECTOR\_SIZE 512**  
809 **HC\_SIZE**  
810 **NR\_HEADS**

811

812 **Zum Beispiel**

- 813 1. Bits fuer Motorsteuerung: DOR, 0x3f2
  - 814 2. Status register des Floppy-Disk-Controllers, FDC\_STATUS, 0x3f4
  - 815 3. Datenregister des Floppy Disk Controllers: FDC\_DATA, 0x3f5
  - 816 4. Register fuer Transferrate: FDC\_RATE
  - 817 5. Ports fuer die unteren 16 Bit der DMA-Adresse DMA\_ADDR
  - 818 6. Status Register 0 : ST0
  - 819 7. Status Register 1 : ST1
  - 820 8. Status Register 2 : ST2
  - 821 9. Status Register 3 : ST3
  - 822 10. Hier meldet der Controller Zylinder: ST\_CYL
  - 823 11. Hier meldet der Controller Koepfe: ST\_HEAD
  - 824 12. Hier meldet der Controller Sektoren: ST\_SEC
  - 825 13. Hier meldet der Controller aktuellen Zylinder: ST\_PCN
- 826 1. I/O-Ports
  - 827 2. Statusregister
  - 828 3. Felder in I/O-Ports

```

829     4. Kommandobytes
830     5. Parameter, wie Sektorgroesse, ...
831     6. Fehlernummern: Positionierung ERR_SEEK

832     floppy_task // Hauptprogramm des Treibers
833     do_rdwrt // Anforderung auf Platte zu lesen oder schreiben
834     dma_setup
835     start_motor
836     stop_motor
837     seek
838     transfer
839     fdc_results
840     fdc_out
841     recalibrate
842     reset
843     clock_mess
844     send_mess

```

## 845 16 kernel/clock.c

```

846     clock_task
847     do_setalarm
848     do_get_time
849     do_set_time
850     do_clock_tick
851     accounting
852     init_clock

```

## 853 17 kernel/tty.c

```

854     tty_task // Hauptroutine
855     do_charint
856     in_char
857     make_break
858     echo
859     chuck
860     do_read
861     rd_chars
862     finish
863     do_write
864     do_ioctl
865     do_cancel
866     tty_reply
867     sigchar
868     keyboard
869     console
870     out_char

```

```
871     scroll_screen
872     flush
873     move_to
874     escape
875     set_6845
876     beep
877     tty_init
878     putc
879     func_key
```

## 880 **18 kernel/system.c**

- 881 1. Zwischenstueck sowohl zwischen Filesystem und kernel
- 882 2. als auch zwischen Speicherverwaltung und Kernel

```
883     SYS_FORK
884     SYS_NEWMAP
885     SYS_EXEC
886     SYS_XIT
887     SYS_GETSP
888     SYS_TIMES
889     SYS_ABORT
890     SYS_SIG
891     SYS_COPY

892     sys_task
893     do_fork
894     do_newmap
895     do_exec
896     do_xit
897     do_getsp
898     do_times
899     do_abort
900     do_sig
901     do_copy
902     cause_sig
903     inform
904     umap
```

## 905 **19 mm**

906 **19.1 mm/const.h**

907 **19.2 mm/glo.h**

908 **19.3 mm/mproc.h**

```
909     EXTERN struct mproc {
910
911     } mproc [NR_PROCS];
```

```

912 EXTERN struct mproc {
913     int mp_...
914     int mp_...
915     int mp_...
916 } mproc [NR_PROCS];

917 EXTERN struct mproc {
918     struct mem_map mp_seg [NR_SEGS];
919     ...
920
921 } mproc [NR_PROCS];

922 EXTERN struct mproc {
923     struct mem_map mp_seg [NR_SEGS];
924     char mp_exitstatus;
925     char mp_sigstatus;
926     int mp_pid;
927     int mp_parent;
928     int mp_procgrp;
929
930     uid mp_realuid;
931     uid mp_effuid;
932     gid mp_realgid;
933     gid mp_effgid;
934
935     unshort mp_ignore;
936     unshort mp_catch;
937     int (*mp_func) ();
938     unsigned mp_flags;
939 } mproc [NR_PROCS];

940 1. mp_seg
941 2. mp_exitstatus
942 3. mp_sigstatus
943 4. mp_pid
944 5. mp_parent
945 6. mp_procgrp
946 7. mp_realuid
947 8. mp_effuid
948 9. mp_realgid
949 10. mp_effgid
950 11. mp_ignore
951 12. mp_catch
952 13. int (*mp_func) ()
953 14. mp_flags

```

954 **19.4** mm/param.h

955 **19.5** mm/type.h

956 **19.6** mm/main.c

957 main

958 get\_work

959 reply

960 mm\_init

961 do\_brk2

962 set\_map

963 **19.7** mm/forkexit.c

964 do\_fork

965 do\_mm\_exit

966 mm\_exit

967 do\_wait

968 cleanup

969 **19.8** mm/exec.c

970 do\_exec

971 read\_header

972 new\_mem

973 patch\_ptr

974 load\_seg

975 **19.9** mm/break.c

976 do\_brk

977 adjust

978 size\_ok

979 stack\_fault

980 **19.10** mm/signal.c

981 do\_signal

982 do\_kill

983 do\_ksig

984 check\_sig

985 sig\_proc

986 do\_alarm

987 set\_alarm

988 do\_pause

989 unpause

990 dump\_core

991 **19.11** mm/getset.c

992 do\_getset



```

993 19.12 mm/alloc.c

994 alloc_mem
995 free_mem
996 del_slot
997 merge
998 max_hole
999 mem_init

1000 PRIVATE struct hole {
1001     phys_clicks h_base;
1002     phys_clicks h_len;
1003     struct hole *h_next;
1004 } hole [NR_HOLES];

1005     Ein Loch besteht aus einem
1006     1. Anfang, Basis
1007     2. Länge
1008     3. Nächster Eintrag in der Freispeicherliste
1009     Es gibt eine Liste mit
1010     1. Zeiger auf Löcher, Liste
1011     2. Zeiger auf unbenutzte Tabelleneinträge, free_slots

1012 PRIVATE struct hole {
1013     phys_clicks h_base;
1014     phys_clicks h_len;
1015     struct hole *h_next;
1016 } hole [NR_HOLES];
1017
1018 PRIVATE struct hole *hole_head;
1019 PRIVATE struct hole *free_slots;
1020
1021 PUBLIC phys_clicks alloc_mem (phys_clicks clicks) {
1022     register struct hole *hp, *prev_ptr;
1023     phys_clicks old_base;
1024
1025
1026     hp = hole_head;
1027     while (hp != NIL_HOLE) {
1028         if (hp -> h_len >= clicks) {
1029             old_base = hp->h_base;
1030             hp->h_base += clicks;
1031             hp->h_len -= clicks;
1032
1033             if (hp->h_len != 0)
1034                 return old_base;
1035             del_slot (prev_ptr, hp);

```

```

1036         return old_base;
1037     }
1038     prev_ptr = hp;
1039     hp = hp->h_next;
1040 }
1041 }
1042
1043 del_slot (struct hole *prev_ptr, struct hole *hp) {
1044     if (hp == hole_head)
1045         hole_head = hp->h_next;
1046     else
1047         prev_ptr->h_next = hp->h_next;
1048     hp->h_next = free_slots;
1049     free_slots = hp;
1050 }
1051

```

#### 1052 Alle funktionen des Speichers

```

1053     main
1054     get_work
1055     reply
1056     mm_init
1057     do_brk2
1058     set_map
1059
1060     do_fork
1061     do_mm_exit
1062     mm_exit
1063     do_wait
1064     cleanup
1065
1066     do_exec
1067     read_header
1068     new_mem
1069     patch_ptr
1070     load_seg
1071
1072     do_brk
1073     adjust
1074     size_ok
1075     stack_fault
1076
1077     do_signal
1078     do_kill
1079     do_ksig
1080     check_sig
1081     sig_proc
1082     do_alarm
1083     set_alarm
1084     do_pause

```

```
1085     unpause
1086     dump_core
1087
1088     do_getset
1089
1090     alloc_mem
1091     free_mem
1092     del_slot
1093     merge
1094     max_hole
1095     mem_init
```