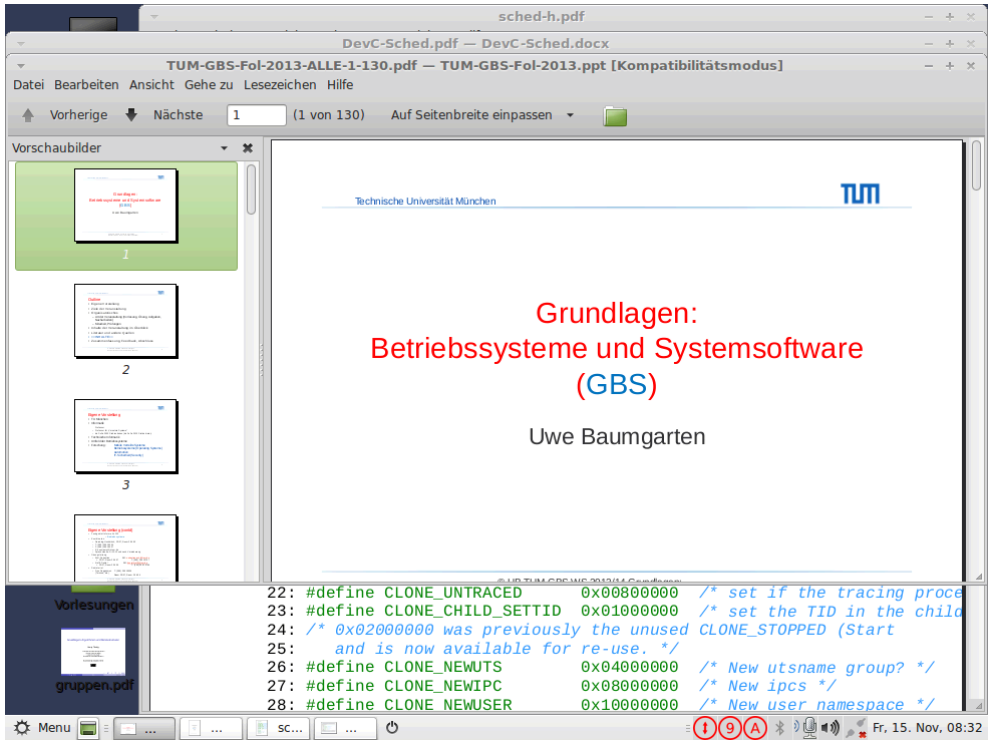
Script **generated by TTT**

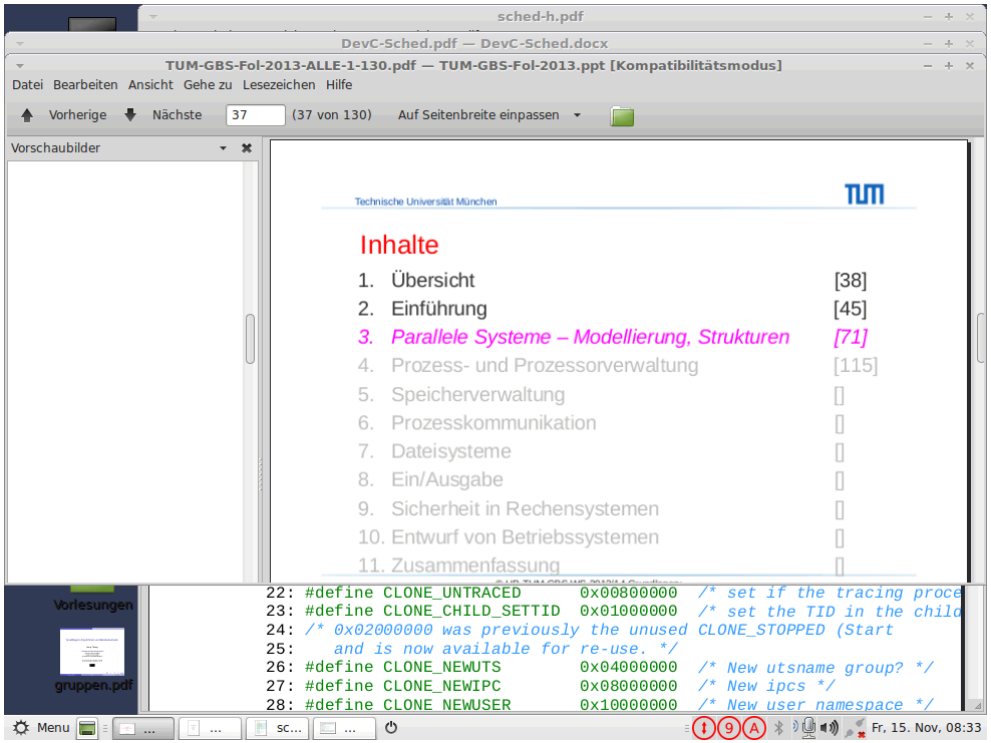
Title: Baumgarten: GBS (15.11.2013)

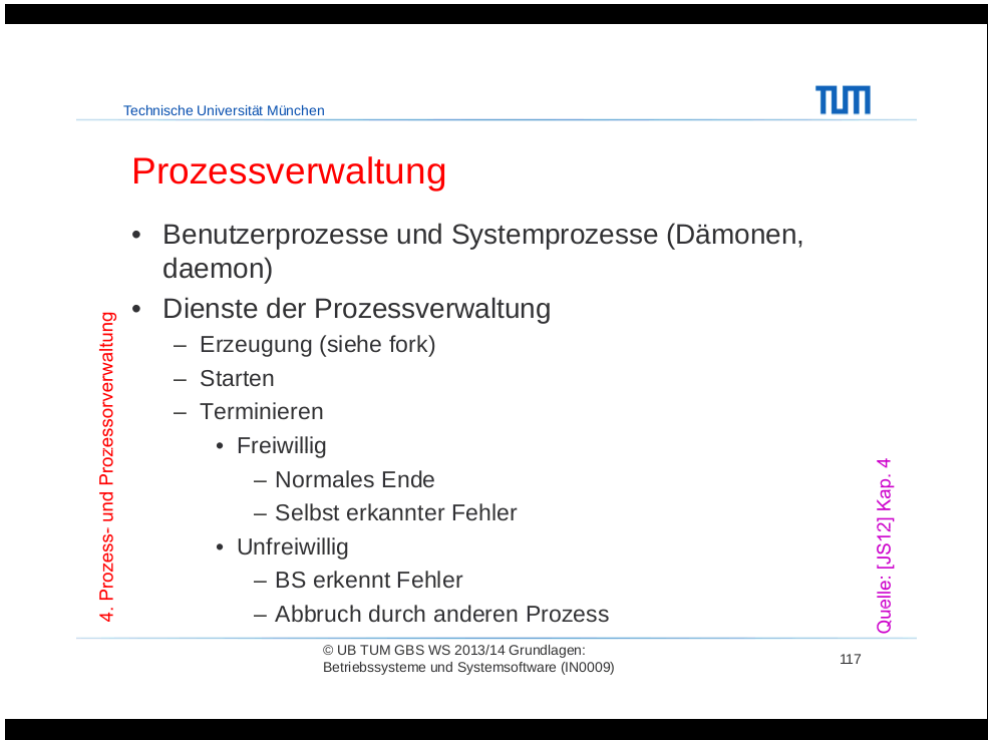
Date: Fri Nov 15 08:32:37 CET 2013

Duration: 89:25 min

Pages: 41







Prozesskontrollblock

- PCB Process Control Block
- Eintrag in Prozesstabelle
- Elemente (beispielhaft)
  - Name
  - Benutzerzuordnung
  - Prozesszustand mit zugeordneter CPU
  - Ereignisse, auf die gewartet wird
  - Priorität
  - Registerinhalte
  - RSW Programmstatuswort (hardwareabhängig)
- Linux „task_struct“
  - include/linux/sched.h

Daten

Multi

Technische Universität München

Prozesskontrollblock

- PCB Process Control Block
- Eintrag in Prozesstabelle
- Elemente (beispielhaft)
  - Name
  - Benutzerzuordnung
  - Prozesszustand mit zugeordneter CPU
  - Ereignisse, auf die gewartet wird
  - Priorität

sched-h.pdf

DevC-Sched.pdf — DevC-Sched.docx

TUM-GBS-Fol-2013-ALLE-1-130.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

```
22: #define CLONE_UNTRACED      0x00800000  /* set if the tracing proce
23: #define CLONE_CHILD_SETTID  0x01000000  /* set the TID in the child
24: /* 0x02000000 was previously the unused CLONE_STOPPED (Start
25:    and is now available for re-use. */
26: #define CLONE_NEWUTS        0x04000000  /* New utsname group? */
27: #define CLONE_NEWIPC        0x08000000  /* New ipcs */
28: #define CLONE_NEWUSER       0x10000000  /* New user namespace */
```

Vorlesungen

gruppen.pdf

Menu

Fr, 15. Nov, 08:46

sched-h.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 1 (1 von 57) Auf Seitenbreite einpassen

Vorschaubilder

```
1: #ifndef _LINUX_SCHED_H
2: #define _LINUX_SCHED_H
3:
4: /*
5:  * cloning flags:
6:  */
7: #define CSIGNAL     0x000000ff  /* signal mask to be sent at exit *
8: #define CLONE_VM    0x00000100  /* set if VM shared between process
9: #define CLONE_FS    0x00000200  /* set if fs info shared between pr
10: #define CLONE_FILES 0x00000400  /* set if open files shared between
11: #define CLONE_SIGHAND   0x00000800  /* set if signal handlers and b
12: #define CLONE_PTRACE    0x00002000  /* set if we want to let tracin
13: #define CLONE_VFORK 0x00004000  /* set if the parent wants the chil
14: #define CLONE_PARENT    0x00008000  /* set if we want to have the s
15: #define CLONE_THREAD    0x00010000  /* Same thread group? */
16: #define CLONE_NEWNS 0x00020000  /* New namespace group? */
17: #define CLONE_SYSVSEM   0x00040000  /* share system V SEM_UNDO sema
18: #define CLONE_SETTLS    0x00080000  /* create a new TLS for the chi
19: #define CLONE_PARENT_SETTID 0x00100000  /* set the TID in the paren
20: #define CLONE_CHILD_CLEARTID    0x00200000  /* clear the TID in the
21: #define CLONE_DETACHED      0x00400000  /* Unused, ignored */
22: #define CLONE_UNTRACED      0x00800000  /* set if the tracing proce
23: #define CLONE_CHILD_SETTID  0x01000000  /* set the TID in the child
24: /* 0x02000000 was previously the unused CLONE_STOPPED (Start
25:    and is now available for re-use. */
26: #define CLONE_NEWUTS        0x04000000  /* New utsname group? */
27: #define CLONE_NEWIPC        0x08000000  /* New ipcs */
28: #define CLONE_NEWUSER       0x10000000  /* New user namespace */
```

Menu sc... Fr, 15. Nov, 08:46

sched-h.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

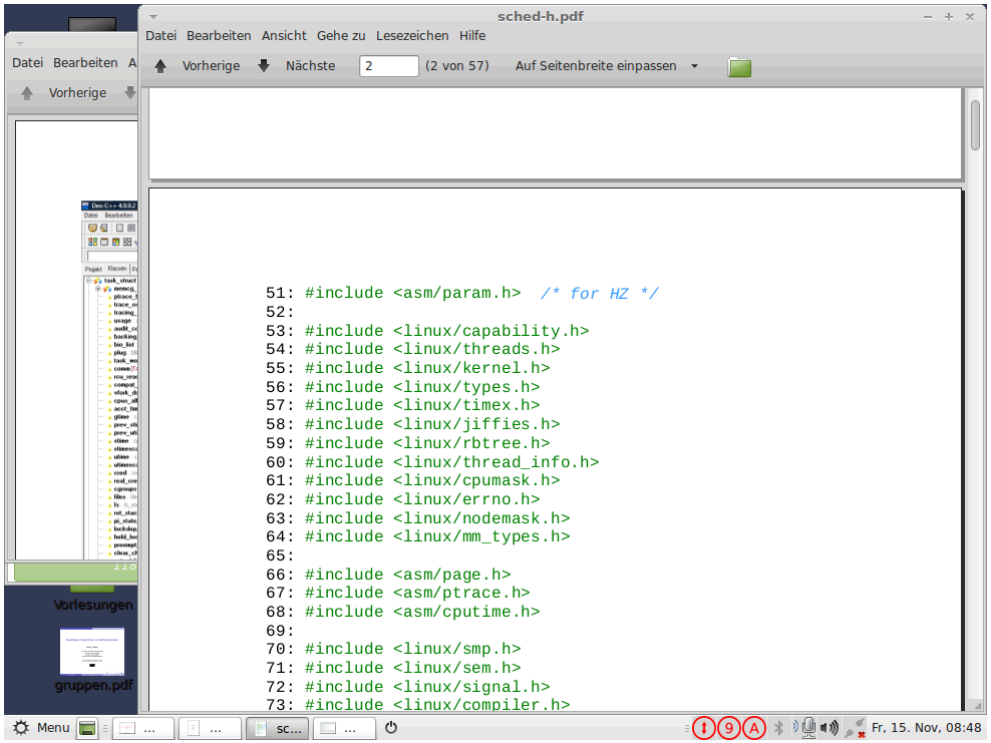
Vorherige Nächste 1 (1 von 57) Auf Seitenbreite einpassen

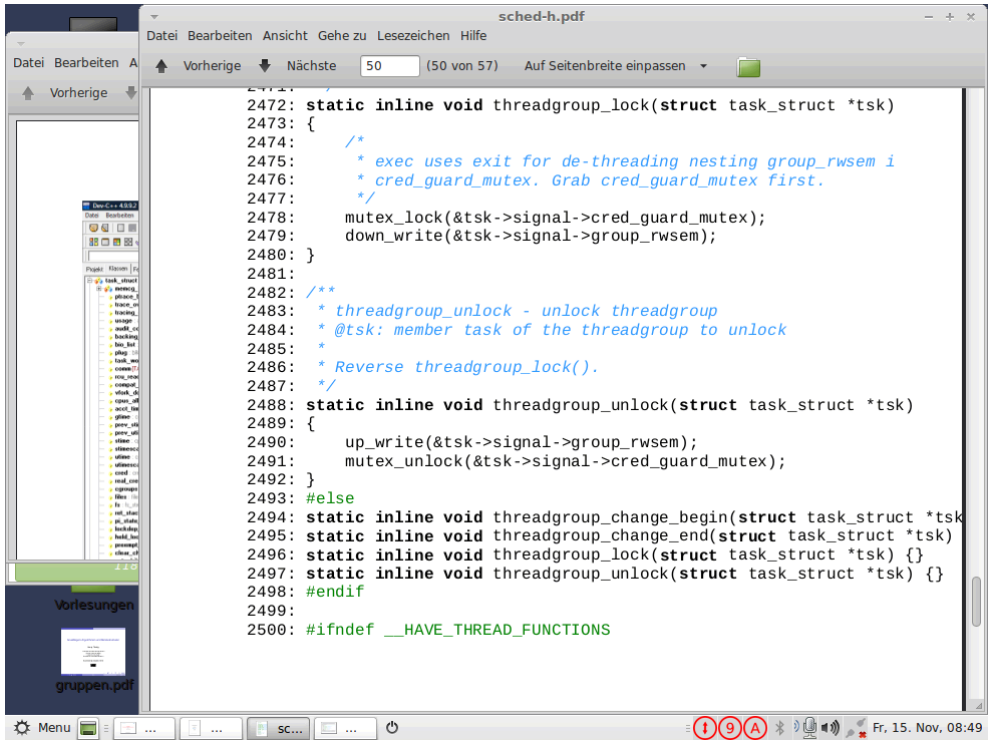
```
1: #ifndef _LINUX_SCHED_H
2: #define _LINUX_SCHED_H
3:
4: /*
5:  * cloning flags:
6:  */
7: #define CSIGNAL     0x000000ff  /* signal mask to be sent at exit *
8: #define CLONE_VM    0x00000100  /* set if VM shared between process
9: #define CLONE_FS    0x00000200  /* set if fs info shared between pr
10: #define CLONE_FILES 0x00000400  /* set if open files shared between
11: #define CLONE_SIGHAND   0x00000800  /* set if signal handlers and b
12: #define CLONE_PTRACE    0x00002000  /* set if we want to let tracin
13: #define CLONE_VFORK 0x00004000  /* set if the parent wants the chil
14: #define CLONE_PARENT    0x00008000  /* set if we want to have the s
15: #define CLONE_THREAD    0x00010000  /* Same thread group? */
16: #define CLONE_NEWNS 0x00020000  /* New namespace group? */
17: #define CLONE_SYSVSEM   0x00040000  /* share system V SEM_UNDO sema
18: #define CLONE_SETTLS    0x00080000  /* create a new TLS for the chi
19: #define CLONE_PARENT_SETTID 0x00100000  /* set the TID in the paren
20: #define CLONE_CHILD_CLEARTID    0x00200000  /* clear the TID in the
21: #define CLONE_DETACHED      0x00400000  /* Unused, ignored */
22: #define CLONE_UNTRACED      0x00800000  /* set if the tracing proce
23: #define CLONE_CHILD_SETTID  0x01000000  /* set the TID in the child
24: /* 0x02000000 was previously the unused CLONE_STOPPED (Start
25:    and is now available for re-use. */
26: #define CLONE_NEWUTS        0x04000000  /* New utsname group? */
27: #define CLONE_NEWIPC        0x08000000  /* New ipcs */
28: #define CLONE_NEWUSER       0x10000000  /* New user namespace */
```

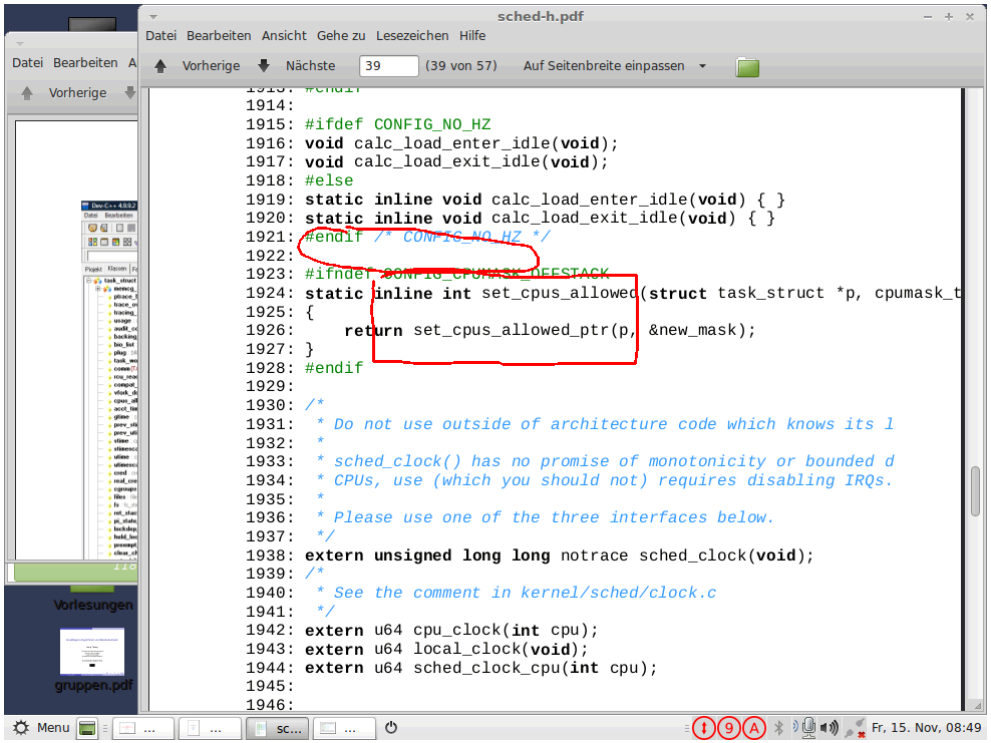
Vorlesungen

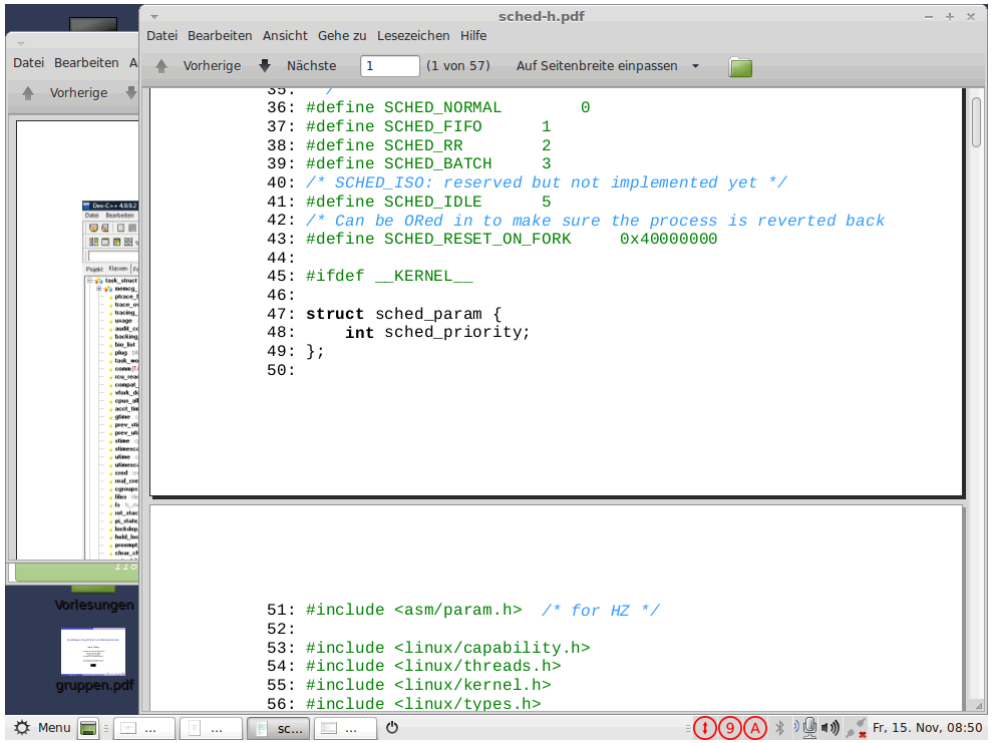
gruppen.pdf

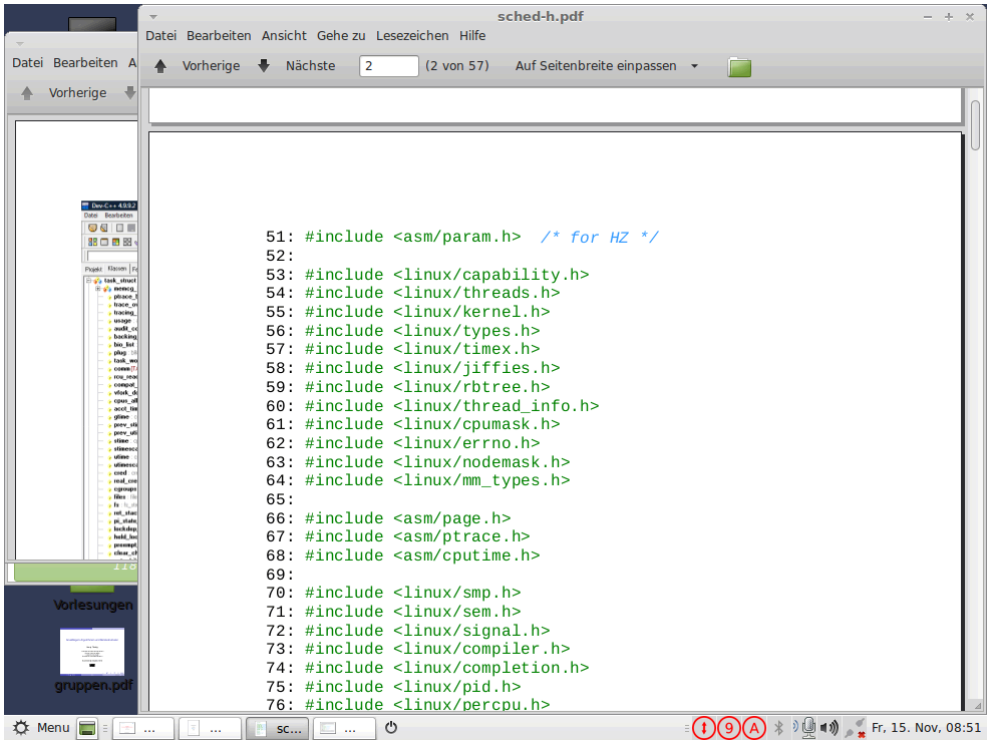
Menu sc... sched-h.pdf Fr, 15. Nov, 08:47

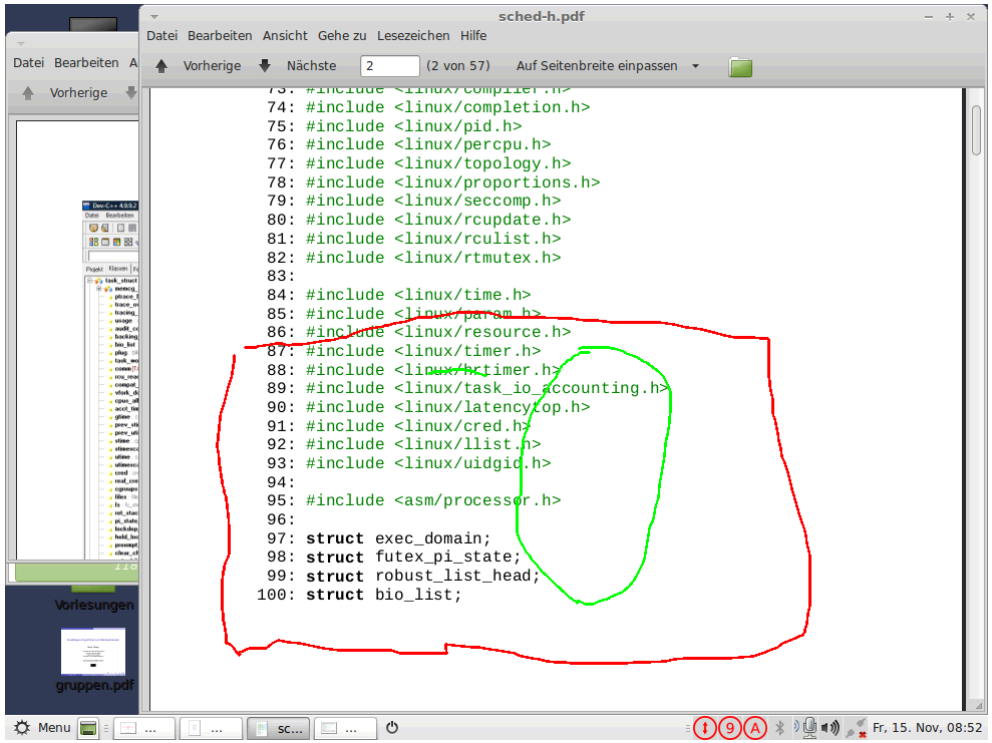

sched-h.pdf
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 2 (2 von 57) Auf Seitenbreite einpassen
51: #include <asm/param.h> /* for HZ */
52:
53: #include <linux/capability.h>
54: #include <linux/threads.h>
55: #include <linux/kernel.h>
56: #include <linux/types.h>
57: #include <linux/timex.h>
58: #include <linux/jiffies.h>
59: #include <linux/rbtree.h>
60: #include <linux/thread_info.h>
61: #include <linux/cpumask.h>
62: #include <linux/errno.h>
63: #include <linux/nodemask.h>
64: #include <linux/mm_types.h>
65:
66: #include <asm/page.h>
67: #include <asm/ptrace.h>
68: #include <asm/cputime.h>
69:
70: #include <linux/smp.h>
71: #include <linux/sem.h>
72: #include <linux/signal.h>
73: #include <linux/compiler.h>
Vorlesungen
gruppen.pdf
Menu
Fr, 15. Nov, 08:48

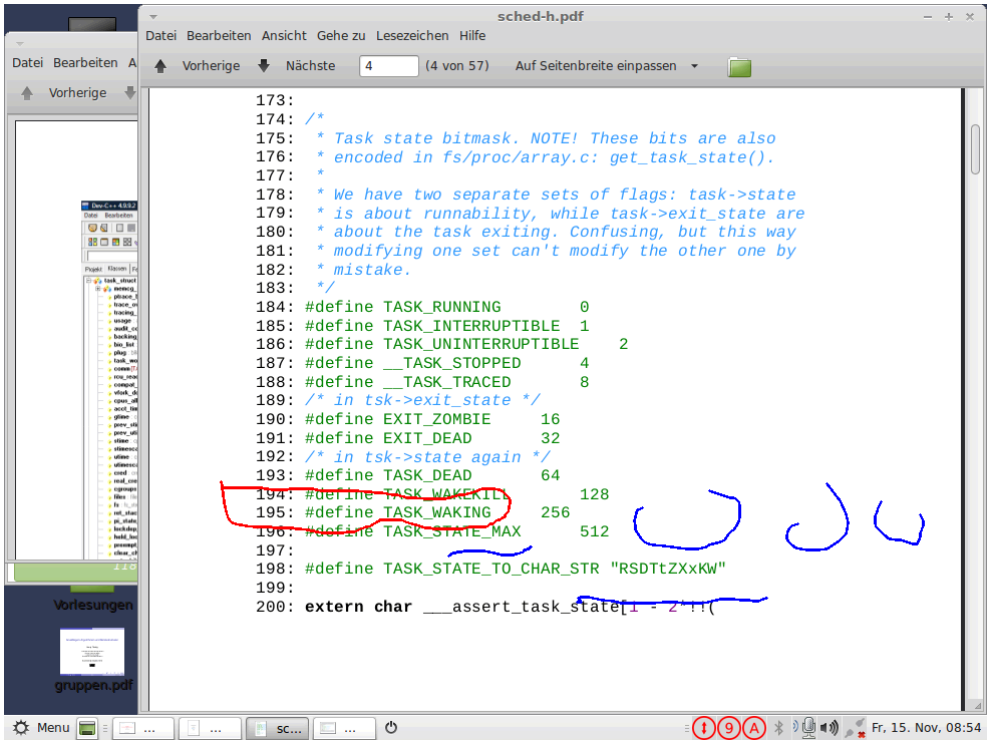

sched-h.pdf
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 50 (50 von 57) Auf Seitenbreite einpassen
2472: static inline void threadgroup_lock(struct task_struct *tsk)
2473: {
2474: /*
2475: * exec uses exit for de-threading nesting group_rwsem i
2476: * cred_guard_mutex. Grab cred_guard_mutex first.
2477: */
2478: mutex_lock(&tsk->signal->cred_guard_mutex);
2479: down_write(&tsk->signal->group_rwsem);
2480: }
2481:
2482: /**
2483: * threadgroup_unlock - unlock threadgroup
2484: * @tsk: member task of the threadgroup to unlock
2485: *
2486: * Reverse threadgroup_lock().
2487: */
2488: static inline void threadgroup_unlock(struct task_struct *tsk)
2489: {
2490: up_write(&tsk->signal->group_rwsem);
2491: mutex_unlock(&tsk->signal->cred_guard_mutex);
2492: }
2493: #else
2494: static inline void threadgroup_change_begin(struct task_struct *tsk
2495: static inline void threadgroup_change_end(struct task_struct *tsk)
2496: static inline void threadgroup_lock(struct task_struct *tsk) {}
2497: static inline void threadgroup_unlock(struct task_struct *tsk) {}
2498: #endif
2499:
2500: #ifndef __HAVE_THREAD_FUNCTIONS
Vorlesungen
gruppen.pdf
Menu
Fr, 15. Nov, 08:49

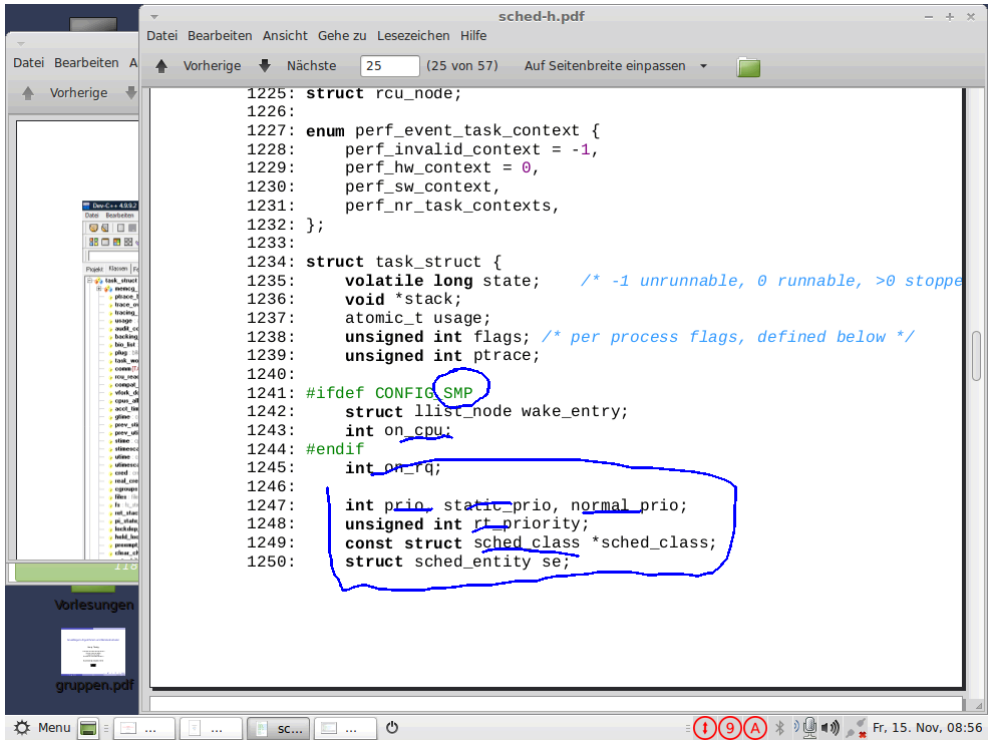

sched-h.pdf
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 39 (39 von 57) Auf Seitenbreite einpassen
1914:
1915: #ifdef CONFIG_NO_HZ
1916: void calc_load_enter_idle(void);
1917: void calc_load_exit_idle(void);
1918: #else
1919: static inline void calc_load_enter_idle(void) { }
1920: static inline void calc_load_exit_idle(void) { }
1921: #endif /* CONFIG_NO_HZ */
1922:
1923: #ifndef CONFIG_CPUMASK_OFFSTACK
1924: static inline int set_cpus_allowed(struct task_struct *p, cpumask_t
1925: {
1926: return set_cpus_allowed_ptr(p, &new_mask);
1927: }
1928: #endif
1929:
1930: /*
1931: * Do not use outside of architecture code which knows its l
1932: *
1933: * sched_clock() has no promise of monotonicity or bounded d
1934: * CPUs, use (which you should not) requires disabling IRQs.
1935: *
1936: * Please use one of the three interfaces below.
1937: */
1938: extern unsigned long long notrace sched_clock(void);
1939: /*
1940: * See the comment in kernel/sched/clock.c
1941: */
1942: extern u64 cpu_clock(int cpu);
1943: extern u64 local_clock(void);
1944: extern u64 sched_clock_cpu(int cpu);
1945:
1946:
Vorlesungen
gruppen.pdf
Menu
Fr, 15. Nov, 08:49

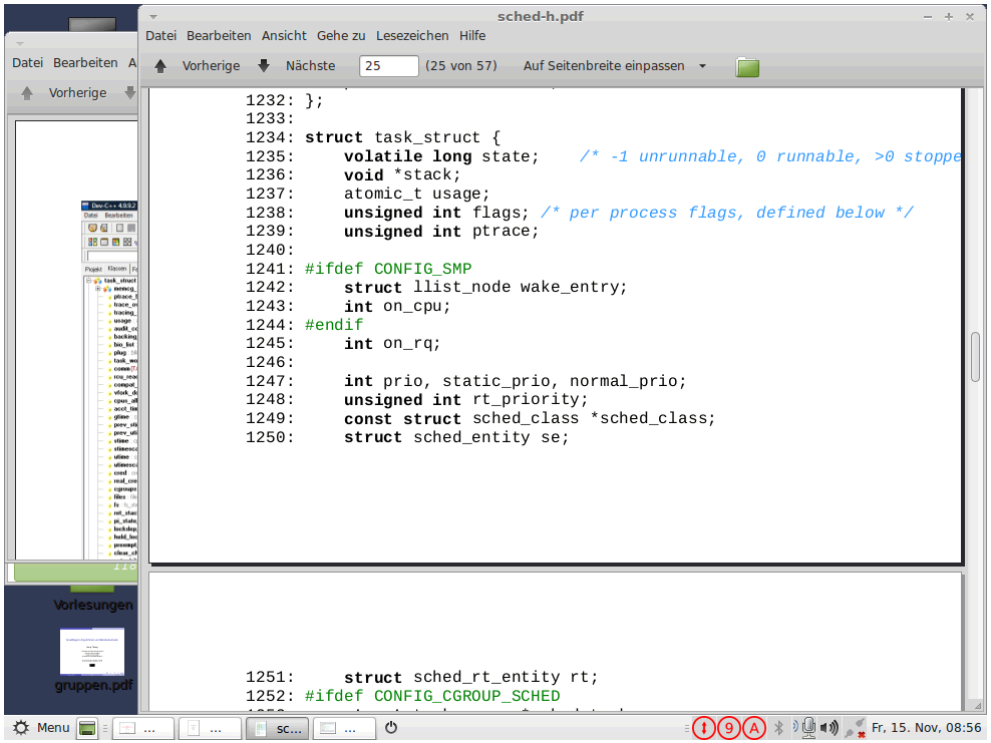

sched-h.pdf
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 1 (1 von 57) Auf Seitenbreite einpassen
36: #define SCHED_NORMAL 0
37: #define SCHED_FIFO 1
38: #define SCHED_RR 2
39: #define SCHED_BATCH 3
40: /* SCHED_ISO: reserved but not implemented yet */
41: #define SCHED_IDLE 5
42: /* Can be ORed in to make sure the process is reverted back
43: #define SCHED_RESET_ON_FORK 0x40000000
44:
45: #ifdef __KERNEL__
46:
47: struct sched_param {
48: int sched_priority;
49: };
50:
51: #include <asm/param.h> /* for HZ */
52:
53: #include <linux/capability.h>
54: #include <linux/threads.h>
55: #include <linux/kernel.h>
56: #include <linux/types.h>
Vorlesungen
gruppen.pdf
Menu
Fr, 15. Nov, 08:50

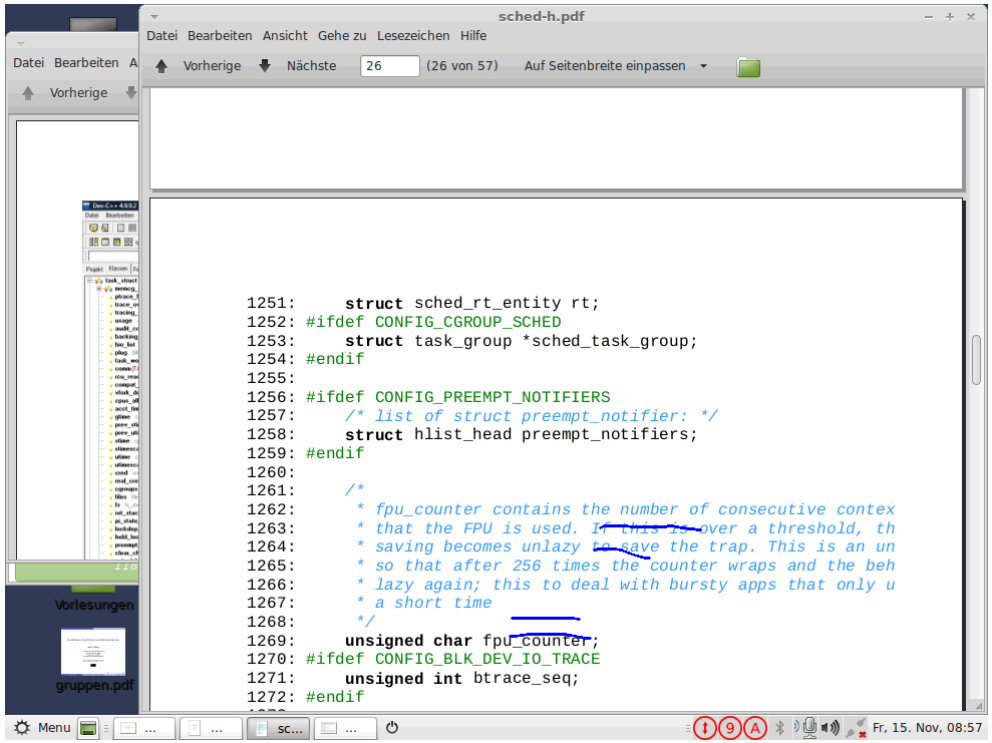

sched-h.pdf
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 2 (2 von 57) Auf Seitenbreite einpassen
51: #include <asm/param.h> /* for HZ */
52:
53: #include <linux/capability.h>
54: #include <linux/threads.h>
55: #include <linux/kernel.h>
56: #include <linux/types.h>
57: #include <linux/timex.h>
58: #include <linux/jiffies.h>
59: #include <linux/rbtree.h>
60: #include <linux/thread_info.h>
61: #include <linux/cpumask.h>
62: #include <linux/errno.h>
63: #include <linux/nodemask.h>
64: #include <linux/mm_types.h>
65:
66: #include <asm/page.h>
67: #include <asm/ptrace.h>
68: #include <asm/cputime.h>
69:
70: #include <linux/smp.h>
71: #include <linux/sem.h>
72: #include <linux/signal.h>
73: #include <linux/compiler.h>
74: #include <linux/completion.h>
75: #include <linux/pid.h>
76: #include <linux/percpu.h>
Vorlesungen
gruppen.pdf
Menu
Fr, 15. Nov, 08:51


sched-h.pdf
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 2 (2 von 57) Auf Seitenbreite einpassen
74: #include <linux/completion.h>
75: #include <linux/pid.h>
76: #include <linux/percpu.h>
77: #include <linux/topology.h>
78: #include <linux/proportions.h>
79: #include <linux/seccomp.h>
80: #include <linux/rcupdate.h>
81: #include <linux/rculist.h>
82: #include <linux/rtmutex.h>
83:
84: #include <linux/time.h>
85: #include <linux/param.h>
86: #include <linux/resource.h>
87: #include <linux/timer.h>
88: #include <linux/hrtimer.h>
89: #include <linux/task_io_accounting.h>
90: #include <linux/latencytop.h>
91: #include <linux/cred.h>
92: #include <linux/llist.h>
93: #include <linux/uidgid.h>
94:
95: #include <asm/processor.h>
96:
97: struct exec_domain;
98: struct futex_pi_state;
99: struct robust_list_head;
100: struct bio_list;
Vorlesungen
gruppen.pdf
Menu
Fr, 15. Nov, 08:52


sched-h.pdf
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 4 (4 von 57) Auf Seitenbreite einpassen
173:
174: /*
175: * Task state bitmask. NOTE! These bits are also
176: * encoded in fs/proc/array.c: get_task_state().
177: *
178: * We have two separate sets of flags: task->state
179: * is about runnability, while task->exit_state are
180: * about the task exiting. Confusing, but this way
181: * modifying one set can't modify the other one by
182: * mistake.
183: */
184: #define TASK_RUNNING 0
185: #define TASK_INTERRUPTIBLE 1
186: #define TASK_UNINTERRUPTIBLE 2
187: #define __TASK_STOPPED 4
188: #define __TASK_TRACED 8
189: /* in tsk->exit_state */
190: #define EXIT_ZOMBIE 16
191: #define EXIT_DEAD 32
192: /* in tsk->state again */
193: #define TASK_DEAD 64
194: #define TASK_WAKEKILL 128
195: #define TASK_WAKING 256
196: #define TASK_STATE_MAX 512
197:
198: #define TASK_STATE_TO_CHAR_STR "RSDTtZXxKW"
199:
200: extern char ___assert_task_state[1 - 2*!!(
Vorlesungen
gruppen.pdf
Menu
Fr, 15. Nov, 08:54


sched-h.pdf
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
Vorherige Nächste 25 (25 von 57) Auf Seitenbreite einpassen
1225: struct rcu_node;
1226:
1227: enum perf_event_task_context {
1228: perf_invalid_context = -1,
1229: perf_hw_context = 0,
1230: perf_sw_context,
1231: perf_nr_task_contexts,
1232: };
1233:
1234: struct task_struct {
1235: volatile long state; /* -1 unrunnable, 0 runnable, >0 stoppe
1236: void *stack;
1237: atomic_t usage;
1238: unsigned int flags; /* per process flags, defined below */
1239: unsigned int ptrace;
1240:
1241: #ifdef CONFIG_SMP
1242: struct llist_node wake_entry;
1243: int on_cpu;
1244: #endif
1245: int on_rq;
1246:
1247: int prio, static_prio, normal_prio;
1248: unsigned int rt_priority;
1249: const struct sched_class *sched_class;
1250: struct sched_entity se;
Vorlesungen
gruppen.pdf
Menu
Fr, 15. Nov, 08:56





Prozesskontrollblock

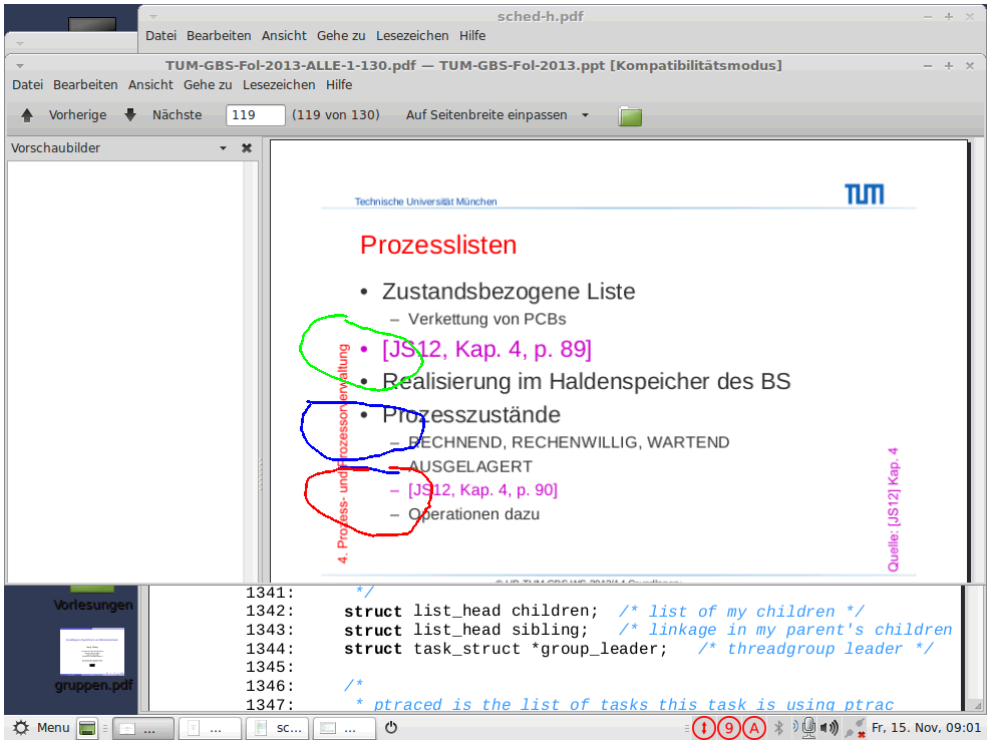
- PCB Process Control Block
- Eintrag in Prozesstabelle
- Elemente (beispielhaft)
  - Name
  - Benutzerzuordnung
  - Prozesszustand mit zugeordneter CPU
  - Ereignisse, auf die gewartet wird
  - Priorität
  - Registerinhalte
  - PSW Programmstatuswort (hardwareabhängig)
- Linux „task_struct“
  - include/linux/sched.h

Quelle: [JS12] Kap. 4

Prozesslisten

- Zustandsbezogene Liste
  - Verkettung von PCBs
- [JS12, Kap. 4, p. 89]
- Realisierung im Haldenspeicher des BS
- Prozesszustände
  - RECHNEND, RECHENWILLIG, WARTEND
  - AUSGELAGERT
  - [JS12, Kap. 4, p. 90]
  - Operationen dazu

Quelle: [JS12] Kap. 4



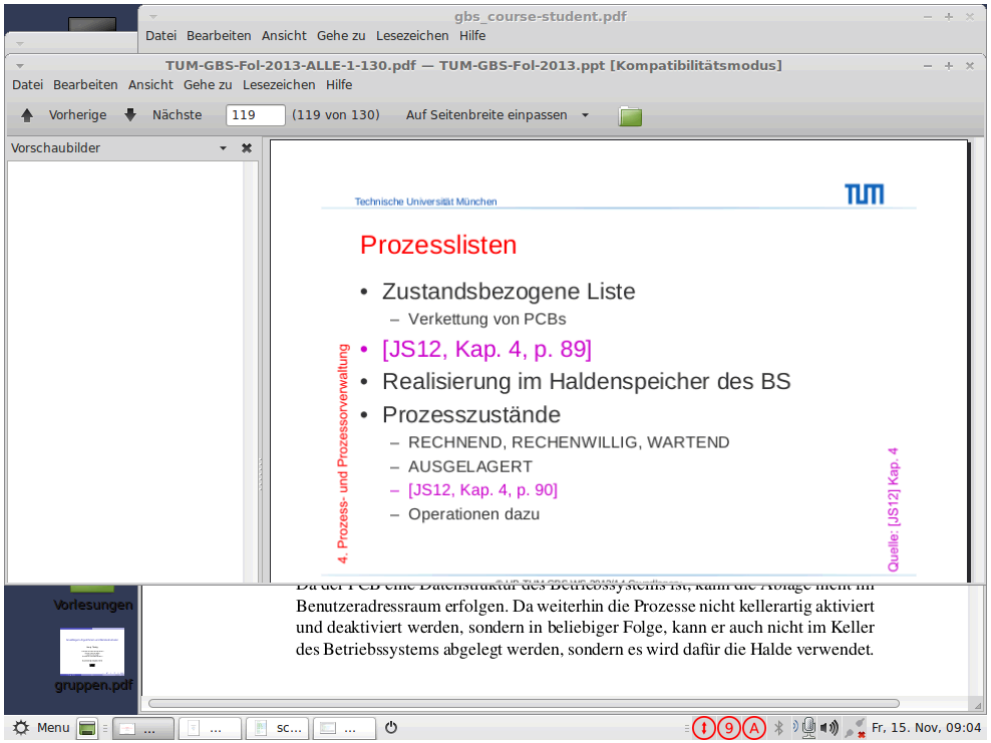
Technische Universität München

Prozesslisten

- Zustandsbezogene Liste
  - Verkettung von PCBs
- [JS12, Kap. 4, p. 89]
- Realisierung im Haldenspeicher des BS
- Prozesszustände
  - RECHNEND, RECHENWILLIG, WARTEND
  - AUSGELAGERT
  - [JS12, Kap. 4, p. 90]
  - Operationen dazu

4. Prozess- und Prozessorverwaltung

Quelle: [JS12] Kap. 4



Technische Universität München

Prozesserzeugung

- Einbindung in Prozesshierarchie
- Vorgehensweise, Alternativen:
  - Windows: Elternprozess veranlasst BS-Aufrufe
  - UNIX:
    - Elternprozess klont sich
    - Kindprozess ändert sich selbst
    - [JS12, Kap. 4, p. 91] Beispielprogramm
    - [JS12, Kap. 4, p. 92] Beispielprogramm
    - Warten auf Prozessende
      - [JS12, Kap. 4, p. 92]
- Vererbung von Betriebsmitteln
  - Ausführung, Ressourcen, Adressraum, Threads

4. Prozess- und Prozessorverwaltung

Quelle: [JS12] Kap. 4

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

TUM-GBS-Fol-2013-ALLE-1-130.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 120 (120 von 130) Auf Seitenbreite einpassen

Vorschaubilder

Technische Universität München

Prozesserzeugung

- Einbindung in Prozesshierarchie
- Vorgehensweise, Alternativen:
  - Windows: Elternprozess veranlasst BS-Aufrufe
  - UNIX:
    - Elternprozess klont sich
    - Kindprozess ändert sich selbst
    - [JS12, Kap. 4, p. 91] Beispielprogramm
    - [JS12, Kap. 4, p. 92] Beispielprogramm
    - Warten auf Prozessende
      - [JS12, Kap. 4, p. 92]
- Vererbung von Betriebsmitteln
  - Ausführung, Ressourcen, Adressraum, Threads

4. Prozess- und Prozessorverwaltung

Quelle: [JS12] Kap. 4

Vorlesungen

gruppen.pdf

Zustandsübergänge sind:

add: ein neu erzeugter Prozess wird zu der Menge der rechenwilligen Prozesse hinzugefügt;

Menu ... sc... ... Fr, 15. Nov, 09:08

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 91 (98 von 228) 150%

Vorschaubilder

116 117 118 119

Kind: 0

– **Beispielprogramm**

```
#include <stdio.h>
int main(int argc, char *argv[]) {
   char *myname = argv[1];
   int cpid = fork();
   if cpid == 0 {
      printf("The child of %s is %d\n", myname, getpid());
      .......
      return (0);
   } else {
      printf("My child is %d\n", cpid);
      /* wird vom Vaterprozess durchlaufen */
      .....
      return(0);
   }
}
```

– Kind hat vieles mit dem Vater gemeinsam:
  liest dieselben Dateien, gleicher Benutzername, benutzt dieselben Daten

– Unix Kind fängt mit dem Code des Vaters an und ändert sich dann
  Systemaufruf exec(): ersetzt das Programmbild des Vaters mit einem anderen.

Beispielprogramm für exec

```
#include <stdio.h>
```

Vorlesungen

gruppen.pdf

Menu ... sc... ... Fr, 15. Nov, 09:09

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 92 (99 von 228) 150%

Vorschaubilder

116 117 118 119


```
int main(int argc, char *argv[]) {
   char *myname = argv[1];
   int cpid = fork();
   if cpid == 0 {
      int rc;
      rc = execl("/bin/ls", "ls", "-l", (char *) 0);
      printf("Fehler bei execl Aufruf: %d\n", rc);
      exit(1)
   } else {
      /* wird vom Vaterprozess durchlaufen */
   }
}
```

– **Prinzipablauf**

Mit der Systemfunktion wait wartet der Vaterprozess auf den Kindprozess.

wait vor Beendigung des Kindes: Vater ist blockiert.

wait nach Beendigung des Kindes: Kind wird nach Beendigung zum Zombieprozess.

Vorlesungen

gruppen.pdf

Menu ... sc... ... Fr, 15. Nov, 09:10

Technische Universität München

Prozesserzeugung

- Einbindung in Prozesshierarchie
- Vorgehensweise, Alternativen:
  - Windows: Elternprozess veranlasst BS-Aufrufe
  - UNIX:
    - Elternprozess klont sich
    - Kindprozess ändert sich selbst
    - [JS12, Kap. 4, p. 91] Beispielprogramm
    - [JS12, Kap. 4, p. 92] Beispielprogramm
    - Warten auf Prozessende
      - [JS12, Kap. 4, p. 92]
- Vererbung von Betriebsmitteln
  - Ausführung, Ressourcen, Adressraum, Threads

4. Prozess- und Prozessorverwaltung

Quelle: [JS12] Kap. 4

Dispatcher

- Realisierung der Zustandsübergänge zwischen RECHNEND und RECHENWILLIG
- Kontextwechsel
- Sichern des Zustandes des aktuell rechnenden Prozesses
- Laden des Zustandes des ausgewählten rechenwilligen Prozesses
- Dispatcher arbeitet im Systemmodus

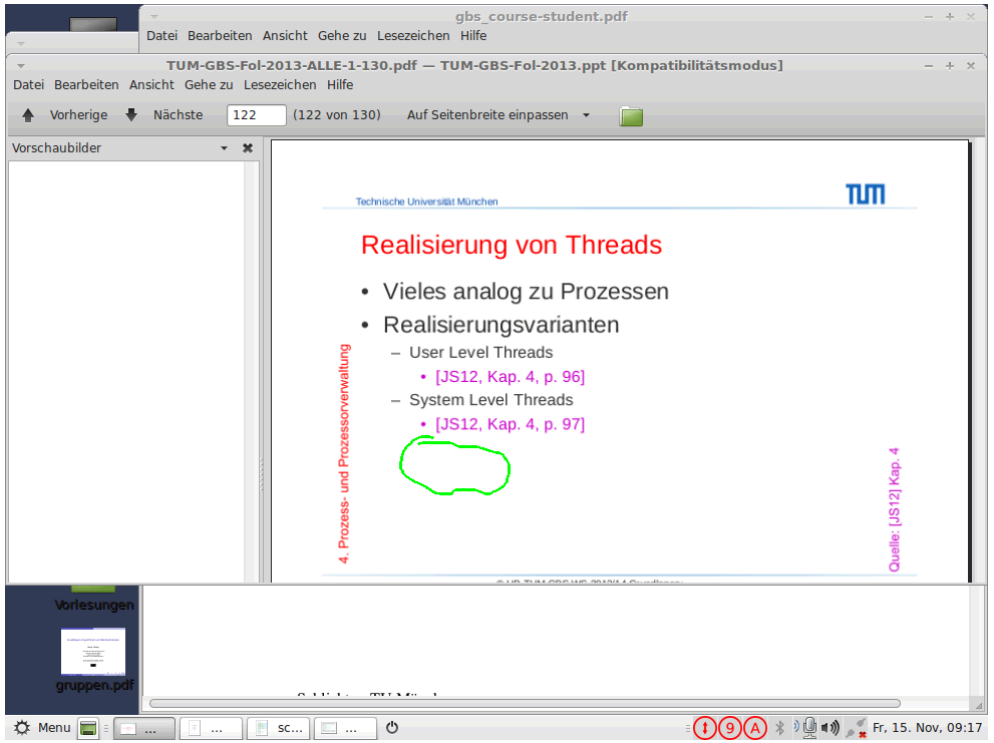
Realisierung von Threads

- Vieles analog zu Prozessen
- Realisierungsvarianten
  - User Level Threads
    - [JS12, Kap. 4, p. 96]
  - System Level Threads
    - [JS12, Kap. 4, p. 97]

Realisierung von Threads

- Vieles analog zu Prozessen
- Realisierungsvarianten
  - User Level Threads
    - [JS12, Kap. 4, p. 96]
  - System Level Threads
    - [JS12, Kap. 4, p. 97]

BS ϕ

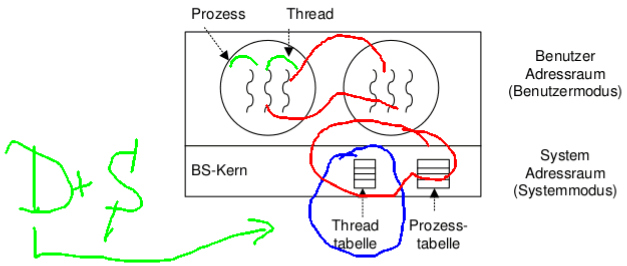


gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 97 (104 von 228) 150%

Vorschaubilder



- Thread-Tabelle speichert Informationen (Register, Zustand, etc.) über Threads.
- Prozessorzuteilung im BS-Kern erfolgt an Threads.

Vorlesungen

gruppen.pdf

Menu ... ... sc... ... Fr, 15. Nov, 09:18

Realisierung von Threads

- Vieles analog zu Prozessen
- Realisierungsvarianten
  - User Level Threads
    - [JS12, Kap. 4, p. 96]
  - System Level Threads
    - [JS12, Kap. 4, p. 97]

Prozessorverwaltung

- Einbindung des Ablaufs
  - Dispatcher
  - Scheduler
  - [JS12, Kap. 4, p. 98]
  - Unterbrechungen

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

TUM-GBS-Fol-2013-ALLE-1-130.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 123 (123 von 130) Auf Seitenbreite einpassen

Vorschaubilder

Technische Universität München

Prozessorverwaltung

- Einbindung des Ablaufs
  - Dispatcher
  - Scheduler
  - [JS12, Kap. 4, p. 98]
  - Unterbrechungen

4. Prozess- und Prozessorverwaltung

Quelle: [JS12] Kap. 4

Vorlesungen

gruppen.pdf

- Thread-Tabelle speichert Informationen (Register, Zustand, etc.) über Threads.
- Prozessorzuteilung im BS-Kern erfolgt an Threads.

Menu ... ... sc... ... Fr, 15. Nov, 09:22

Scheduling - Überblick

- Aufgabe des Schedulers
  - Auswahl des nächsten Prozesses (Threads)
- Kriterien
- Strategien
- Bewertung

Quelle: [JS12] Kap. 4

Scheduling - Kriterien

- Fairness
- Effizienz, Prozessorauslastung
- Antwortzeit (insbs. bei Dialogverarbeitung)
- Wartezeit (insbs. bei Stapelverarbeitung)
- Ausführungszeit
- Abschlusszeit (insbs. bei Realverarbeitung)
- Durchsatz

Quelle: [JS12] Kap. 4

Scheduling – Strategien

- Klassen
  - Unterbrechend (preemptive), Nicht-unterbrechend (nonpreemptive)
  - Mit/ohne Prioritäten
  - Mit/ohne Deadlines
- Beispiele für Strategien
  - Zeitscheibenstrategie
    - Round Robin, Quantum
  - Prioritätsstrategien
    - Statiche Prioritäten
    - Dynamische Prioritäten
  - FCFS (First-Come-First-Served)
  - SJF (Shortest Job First)
    - SRPT (Shorted Remainig Processing Time)
  - EDF (Earliest Deadline First), RMS (Rate Monotonic Scheduling)

Quelle: [JS12] Kap. 4

Exkurs: Scheduling – Strategien – Bewertung

- CPU als Bediener-Modell
- Prozesse als Aufträge zur Berechnung
- Charakterisierung der Aufträge
  - Ankunftszeit
  - Bedienzeit
  - Abgangszeit
  - Wartezeit
  - Verweilzeit
- Beispiele

Scheduling - Kriterien

- Fairness
- Effizienz, Prozessorauslastung
- Antwortzeit (insbs. bei Dialogverarbeitung)
- Wartezeit (insbs. bei Stapelverarbeitung)
- Ausführungszeit
- Abschlusszeit (insbs. bei Realverarbeitung)
- Durchsatz

gbs_course-student.pdf
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe
TUM-GBS-Fol-2013-ALLE-1-130.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]
Technische Universität München
Scheduling - Kriterien
Fairness
Effizienz, Prozessorauslastung
Antwortzeit (insbs. bei Dialogverarbeitung)
Wartezeit (insbs. bei Stapelverarbeitung)
Ausführungszeit
Abschlusszeit (insbs. bei Realverarbeitung)
nd Prozessorverwaltung
Kap. 4
Vorlesungen
gruppen.pdf
4.3.1 Kriterien
Der Scheduler wählt aus der Menge der rechenwilligen Prozesse den nächsten
Menu
Fr, 15. Nov, 10:01