```
struct super_block {
  struct list_head  s_list;
  ...
  unsigned long     s_blocksize;
  loff_t      s_maxbytes;
  struct file_system_type *s_type;
  const struct super_operations   *s_op;
  unsigned long     s_flags;
  unsigned long     s_magic;
  struct dentry     *s_root;
  struct rw_semaphore     s_umount;
  int   s_count;
  char  s_id[32];
  const struct dentry_operations *s_d_op;
  struct list_lru   s_dentry_lru;
  struct list_lru   s_inode_lru;
  struct list_head  s_inodes;
  ...
} __randomize_layout;
```

```
struct super_operations {
  struct inode *(*alloc_inode)(struct super_block *sb);
  void (*destroy_inode)(struct inode *);
  void (*free_inode)(struct inode *);
  void (*dirty_inode) (struct inode *, int flags);
  int (*write_inode) (struct inode *, struct writeback_control *wbc);
  int (*drop_inode) (struct inode *);
  void (*put_super) (struct super_block *);
  int (*statfs) (struct dentry *, struct kstatfs *);
  ...
};
```

```
struct inode {
  umode_t     i_mode;
  kuid_t      i_uid;
  kgid_t      i_gid;
  unsigned int      i_flags;
  const struct inode_operations   *i_op;
  struct super_block      *i_sb;
  unsigned long     i_ino;
  union {
    const unsigned int i_nlink;
    unsigned int __i_nlink;
  };
  dev_t       i_rdev;
  loff_t      i_size;
  struct timespec64       __i_atime;
  struct timespec64       __i_mtime;
  struct timespec64       __i_ctime;
  spinlock_t  i_lock;
  unsigned short    i_bytes;
  u8    i_blkbits;
  blkcnt_t    i_blocks;
  struct hlist_node       i_hash;
  atomic64_t  i_version;
  atomic_t    i_count;
  ...
} __randomize_layout;
```

```
struct inode_operations {
  struct dentry * (*lookup) (struct inode *,struct dentry *, unsigned int);
  int (*permission) (struct mnt_idmap *, struct inode *, int);
  int (*create) (struct mnt_idmap *, struct inode *,struct dentry *,
     umode_t, bool);
  int (*link) (struct dentry *,struct inode *,struct dentry *);
  int (*unlink) (struct inode *,struct dentry *);
  int (*symlink) (struct mnt_idmap *, struct inode *,struct dentry *,
```

```
      const char *);
  int (*mkdir) (struct mnt_idmap *, struct inode *,struct dentry *,
    umode_t);
  int (*rmdir) (struct inode *,struct dentry *);
  int (*mknod) (struct mnt_idmap *, struct inode *,struct dentry *,
    umode_t,dev_t);
  int (*rename) (struct mnt_idmap *, struct inode *, struct dentry *,
     struct inode *, struct dentry *, unsigned int);
  int (*atomic_open)(struct inode *, struct dentry *,
   struct file *, unsigned open_flag,
   umode_t create_mode);
  ...
} ____cacheline_aligned;
```

```
struct dentry { // v6.10-rc3
...
  struct hlist_bl_node d_hash;
  struct dentry *d_parent;
  struct qstr d_name;
  struct inode *d_inode;
  unsigned char d_iname[DNAME_INLINE_LEN];
  const struct dentry_operations *d_op;
  struct super_block *d_sb;
  union {
    struct list_head d_lru;
    wait_queue_head_t *d_wait;
  };
  struct hlist_node d_sib;
  struct hlist_head d_children;
...
};
struct dentry { // v2.6.10
  atomic_t     d_count;
  unsigned int       d_flags;
  spinlock_t         d_lock;
  struct inode       *d_inode;
  struct dentry      *d_parent;
  struct qstr        d_name;
  struct list_head   d_lru;
  struct list_head   d_child;
  struct list_head   d_subdirs;
  struct list_head   d_alias;
  unsigned long      d_time;
  struct dentry_operations *d_op;
  struct super_block       *d_sb;
  void         *d_fsdata;
  struct rcu_head    d_rcu;
  struct dcookie_struct    *d_cookie;
  struct hlist_node  d_hash;
  int d_mounted;
  unsigned char      d_iname[DNAME_INLINE_LEN_MIN];
};
```

```
struct dentry_operations {
  int (*d_revalidate)(struct dentry *, unsigned int);
  int (*d_hash)(const struct dentry *, struct qstr *);
  int (*d_compare)(const struct dentry *,
                   unsigned int, const char *, const struct qstr *);
  int (*d_init)(struct dentry *);
  void (*d_release)(struct dentry *);
  char *(*d_dname)(struct dentry *, char *, int);
  ...
} ____cacheline_aligned;
```

```
struct kmem_cache {
#ifndef CONFIG_SLUB_TINY
        struct kmem_cache_cpu __percpu *cpu_slab;
#endif
        /* Used for retrieving partial slabs, etc. */
        slab_flags_t flags;
        unsigned long min_partial;
        unsigned int size;              /* Object size including metadata */
        unsigned int object_size;       /* Object size without metadata */
        struct reciprocal_value reciprocal_size;
        unsigned int offset;            /* Free pointer offset */
#ifdef CONFIG_SLUB_CPU_PARTIAL
        /* Number of per cpu partial objects to keep around */
        unsigned int cpu_partial;
        /* Number of per cpu partial slabs to keep around */
        unsigned int cpu_partial_slabs;
#endif
        struct kmem_cache_order_objects oo;
        /* Allocation and freeing of slabs */
        struct kmem_cache_order_objects min;
        gfp_t allocflags;               /* gfp flags to use on each alloc */
        int refcount;                   /* Refcount for slab cache destroy */
        void (*ctor)(void *object);     /* Object constructor */
        unsigned int inuse;             /* Offset to metadata */
        unsigned int align;             /* Alignment */
        unsigned int red_left_pad;      /* Left redzone padding size */
        const char *name;               /* Name (only for display!) */
        struct list_head list;          /* List of slab caches */
#ifdef CONFIG_SYSFS
        struct kobject kobj;            /* For sysfs */
#endif
#ifdef CONFIG_SLAB_FREELIST_HARDENED
        unsigned long random;
#endif
#ifdef CONFIG_NUMA
        unsigned int remote_node_defrag_ratio;
#endif
#ifdef CONFIG_SLAB_FREELIST_RANDOM
        unsigned int *random_seq;
#endif
#ifdef CONFIG_KASAN_GENERIC
        struct kasan_cache kasan_info;
#endif
#ifdef CONFIG_HARDENED_USERCOPY
        unsigned int useroffset;        /* Usercopy region offset */
        unsigned int usersize;          /* Usercopy region size */
#endif
        struct kmem_cache_node *node[MAX_NUMNODES];
};
```

```
struct socket {
  socket_state      state;
  short       type;
  unsigned long     flags;
  struct file       *file;
  struct sock       *sk;
  const struct proto_ops  *ops;
  struct socket_wq  wq;
};
```

```
struct sockaddr {
  sa_family_t     sa_family;      /* address family, AF_xxx       */
  union {
    char sa_data_min[14];           /* Minimum 14 bytes of protocol address */
    DECLARE_FLEX_ARRAY(char, sa_data);
  };
};
```

```
struct sockaddr_in {
  __kernel_sa_family_t  sin_family;     /* Address family               */
  __be16                sin_port;       /* Port number                  */
  struct in_addr        sin_addr;       /* struct in_addr {__be32  s_addr;};*/
            */

  /* Pad to size of `struct sockaddr'. */
  unsigned char         __pad[__SOCK_SIZE__ - sizeof(short int) -
                        sizeof(unsigned short int) - sizeof(struct in_addr)];
};
```

```
struct sockaddr_in6 {
  unsigned short int      sin6_family;    /* AF_INET6 */
  __be16                  sin6_port;      /* Transport layer port # */
  __be32                  sin6_flowinfo;  /* IPv6 flow information */
  struct in6_addr         sin6_addr;      /* IPv6 address */
  __u32                   sin6_scope_id;  /* scope id (new in RFC2553) */
};
```

```
struct file {
  ...
  spinlock_t        f_lock;
  fmode_t           f_mode;
  atomic_long_t     f_count;
  struct mutex      f_pos_lock;
  loff_t            f_pos;
  unsigned int      f_flags;
  struct fown_struct      f_owner;
  struct path             f_path;
  struct inode            *f_inode;       /* cached value */
  const struct file_operations    *f_op;
  u64         f_version;
  struct address_space    *f_mapping;
  ...
} __randomize_layout
  __attribute__((aligned(4)));  /* lest something weird decides that 2 is OK */
```

```
struct path {
  struct vfsmount *mnt;
  struct dentry *dentry;
} __randomize_layout;
```

```
struct file_operations {
  struct module *owner;
  loff_t (*llseek) (struct file *, loff_t, int);
  ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
  ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
  int (*mmap) (struct file *, struct vm_area_struct *);
  int (*open) (struct inode *, struct file *);
  int (*flush) (struct file *, fl_owner_t id);
  int (*release) (struct inode *, struct file *);
  ...
} __randomize_layout;
```

```
struct file_system_type {
  const char *name;
  int fs_flags;
#define FS_REQUIRES_DEV         1
#define FS_USERNS_MOUNT         8       /* Can be mounted by userns root */
  struct dentry *(*mount) (struct file_system_type *, int,
                           const char *, void *);
  void (*kill_sb) (struct super_block *);
  struct module *owner;
  struct file_system_type *next;
```

```
  struct hlist_head fs_supers;
  struct lock_class_key s_lock_key;
};
```

```
struct fs_struct {
  int users;
  spinlock_t lock;
  seqcount_spinlock_t seq;
  int umask;
  int in_exec;
  struct path root, pwd;
} __randomize_layout;
```

```
struct files_struct {
  atomic_t count;
  bool resize_in_progress;
  wait_queue_head_t resize_wait;
  struct fdtable __rcu *fdt;
  struct fdtable fdtab;
  spinlock_t file_lock ____cacheline_aligned_in_smp;
  unsigned int next_fd;
  unsigned long close_on_exec_init[1];
  unsigned long open_fds_init[1];
  unsigned long full_fds_bits_init[1];
  struct file __rcu *fd_array[NR_OPEN_DEFAULT];
};
```

```
struct tasklet_struct
{
  struct tasklet_struct *next;
  unsigned long state;
  atomic_t count;
  bool use_callback;
  union {
    void (*func)(unsigned long data);
    void (*callback)(struct tasklet_struct *t);
  };
  unsigned long data;
};

enum
{
  TASKLET_STATE_SCHED,    /* Tasklet is scheduled for execution */
  TASKLET_STATE_RUN       /* Tasklet is running (SMP only) */
};
```

```
struct workqueue_struct {
  struct list_head        pwqs;           /* WR: all pwqs of this wq */
  struct list_head        list;           /* PR: list of all workqueues */
  struct worker           *rescuer;       /* MD: rescue worker */
  char                    name[WQ_NAME_LEN]; /* I: workqueue name */
  struct pool_workqueue __percpu __rcu **cpu_pwq; /* I: per-cpu pwqs */
};
```

```
struct work_struct {
  atomic_long_t data;
  struct list_head entry;
  work_func_t func;
#ifdef CONFIG_LOCKDEP
  struct lockdep_map lockdep_map;
#endif
};
```

```
struct pool_workqueue {
```

```
  struct worker_pool      *pool;          /* I: the associated pool */
  struct workqueue_struct *wq;            /* I: the owning workqueue */
  int                     work_color;     /* L: current color */
  int                     flush_color;    /* L: flushing color */
  int                     refcnt;         /* L: reference count */
  int                     nr_in_flight[WORK_NR_COLORS];
                                          /* L: nr of in_flight works */
  bool                    plugged;        /* L: execution suspended */
  int                     nr_active;      /* L: nr of active works */
  struct list_head        inactive_works; /* L: inactive works */
  struct list_head        pending_node;   /* LN: node on wq_node_nr_active->pending_pwqs
*/
  struct list_head        pwqs_node;      /* WR: node on wq->pwqs */
  struct list_head        mayday_node;    /* MD: node on wq->maydays */
  u64                     stats[PWQ_NR_STATS];
  struct kthread_work     release_work;
  struct rcu_head         rcu;
} __aligned(1 << WORK_STRUCT_PWQ_SHIFT);
```

```
struct worker_pool {
  raw_spinlock_t          lock;           /* the pool lock */
  int                     cpu;            /* I: the associated cpu */
  int                     node;           /* I: the associated node ID */
  int                     id;             /* I: pool ID */
  unsigned int            flags;          /* L: flags */
  unsigned long           watchdog_ts;    /* L: watchdog timestamp */
  bool                    cpu_stall;      /* WD: stalled cpu bound pool */
  int                     nr_running;
  struct list_head        worklist;       /* L: list of pending works */
  int                     nr_workers;     /* L: total number of workers */
  int                     nr_idle;        /* L: currently idle workers */
  struct list_head        idle_list;      /* L: list of idle workers */
  struct timer_list       idle_timer;     /* L: worker idle timeout */
  struct work_struct      idle_cull_work; /* L: worker idle cleanup */
  struct timer_list       mayday_timer;   /* L: SOS timer for workers */
  /* a workers is either on busy_hash or idle_list, or the manager */
  DECLARE_HASHTABLE(busy_hash, BUSY_WORKER_HASH_ORDER);
                                          /* L: hash of busy workers */
  struct worker           *manager;       /* L: purely informational */
  struct list_head        workers;        /* A: attached workers */
  struct list_head        dying_workers;  /* A: workers about to die */
  struct completion       *detach_completion; /* all workers detached */
  struct ida              worker_ida;     /* worker IDs for task name */
  struct workqueue_attrs  *attrs;         /* I: worker attributes */
  struct hlist_node       hash_node;      /* PL: unbound_pool_hash node */
  int                     refcnt;         /* PL: refcnt for unbound pools */
  struct rcu_head         rcu;
};
```

```
struct cpu_workqueue_struct {
  spinlock_t lock;
  long remove_sequence;
  long insert_sequence;
  struct list_head worklist;
  wait_queue_head_t more_work;
  wait_queue_head_t work_done;
  struct workqueue_struct *wq;
  task_t *thread;
  int run_depth;
} __cacheline_aligned;
```

```
typedef u32 __kernel_dev_t;
typedef __kernel_dev_t dev_t;
```

```
struct device_driver {
  const char              *name;
```

```
  const struct bus_type   *bus;
  struct module           *owner;
  ...
  const struct of_device_id       *of_match_table;
  int (*probe) (struct device *dev);
  int (*remove) (struct device *dev);
  void (*shutdown) (struct device *dev);
  ...
};
```

```
struct device {
  struct kobject kobj;
  struct device           *parent;
  const char              *init_name; /* initial name of the device */
  const struct device_type *type;
  const struct bus_type   *bus;   /* type of bus device is on */
  struct device_driver *driver;   /* which driver has allocated this
                                     device */
  u64             *dma_mask;      /* dma mask (if dma'able device) */
  u64             bus_dma_limit;  /* upstream dma constraint */
#ifdef CONFIG_NUMA
  int             numa_node;      /* NUMA node this device is close to */
#endif
  dev_t                   devt;   /* dev_t, creates the sysfs "dev" */
  ...
};
```

```
struct usb_driver {
  const char *name;
  int (*probe) (struct usb_interface *intf,
                const struct usb_device_id *id);
  void (*disconnect) (struct usb_interface *intf);
  int (*unlocked_ioctl) (struct usb_interface *intf, unsigned int code,
                         void *buf);
  int (*suspend) (struct usb_interface *intf, pm_message_t message);
  int (*resume) (struct usb_interface *intf);
  int (*reset_resume)(struct usb_interface *intf);
  int (*pre_reset)(struct usb_interface *intf);
  int (*post_reset)(struct usb_interface *intf);
  const struct usb_device_id *id_table;
  const struct attribute_group **dev_groups;
  struct usb_dynids dynids;
  struct device_driver driver;
  unsigned int no_dynamic_id:1;
  unsigned int supports_autosuspend:1;
  unsigned int disable_hub_initiated_lpm:1;
  unsigned int soft_unbind:1;
};
```

```
struct usb_device_driver {
  const char *name;
  bool (*match) (struct usb_device *udev);
  int (*probe) (struct usb_device *udev);
  void (*disconnect) (struct usb_device *udev);
  int (*suspend) (struct usb_device *udev, pm_message_t message);
  int (*resume) (struct usb_device *udev, pm_message_t message);
  int (*choose_configuration) (struct usb_device *udev);
  const struct attribute_group **dev_groups;
  struct device_driver driver;
  const struct usb_device_id *id_table;
  unsigned int supports_autosuspend:1;
  unsigned int generic_subclass:1;
};
```

```
struct softirq_action
{
```

```
  void  (*action)(struct softirq_action *);
};
```

```
struct proc_dir_entry {
  atomic_t in_use;
  refcount_t refcnt;
  struct list_head pde_openers;   /* who did ->open, but not ->release */
  /* protects ->pde_openers and all struct pde_opener instances */
  spinlock_t pde_unload_lock;
  struct completion *pde_unload_completion;
  const struct inode_operations *proc_iops;
  union {
    const struct proc_ops *proc_ops;
    const struct file_operations *proc_dir_ops;
  };
  const struct dentry_operations *proc_dops;
  union {
    const struct seq_operations *seq_ops;
    int (*single_show)(struct seq_file *, void
  };
  proc_write_t write;
  void *data;
  unsigned int state_size;
  unsigned int low_ino;
  nlink_t nlink;
  kuid_t uid;
  kgid_t gid;
  loff_t size;
  struct proc_dir_entry *parent;
  struct rb_root subdir;
  struct rb_node subdir_node;
  char *name;
  umode_t mode;
  u8 flags;
  u8 namelen;
  char inline_name[];
} __randomize_layout;
```

```
struct proc_ops {
  unsigned int proc_flags;
  int     (*proc_open)(struct inode *, struct file *);
  ssize_t (*proc_read)(struct file *, char __user *, size_t, loff_t *);
  ssize_t (*proc_read_iter)(struct kiocb *, struct iov_iter *);
  ssize_t (*proc_write)(struct file *, const char __user *, size_t, loff_t *);
  /* mandatory unless nonseekable_open() or equivalent is used */
  loff_t  (*proc_lseek)(struct file *, loff_t, int);
  int     (*proc_release)(struct inode *, struct file *);
  __poll_t (*proc_poll)(struct file *, struct poll_table_struct *);
  long    (*proc_ioctl)(struct file *, unsigned int, unsigned long);
#ifdef CONFIG_COMPAT
  long    (*proc_compat_ioctl)(struct file *, unsigned int, unsigned long);
#endif
  int     (*proc_mmap)(struct file *, struct vm_area_struct *);
  unsigned long (*proc_get_unmapped_area)(struct file *, unsigned long, unsigned long,
unsigned long, unsigned long);
} __randomize_layout;
```

```
struct seq_file {
  char *buf;
  size_t size;
  size_t from;
  size_t count;
  size_t pad_until;
  loff_t index;
  loff_t read_pos;
  struct mutex lock;
  const struct seq_operations *op;
```

```
  int poll_event;
  const struct file *file;
  void *private;
};
```

```
struct seq_operations {
  void * (*start) (struct seq_file *m, loff_t *pos);
  void (*stop) (struct seq_file *m, void *v);
  void * (*next) (struct seq_file *m, void *v, loff_t *pos);
  int (*show) (struct seq_file *m, void *v);
};
```

```
struct epoll_event {
  uint32_t      events;  /* Epoll events */
  epoll_data_t  data;    /* User data variable */
};

union epoll_data {
  void     *ptr;
  int       fd;
  uint32_t  u32;
  uint64_t  u64;
};

typedef union epoll_data  epoll_data_t
```

STRUCT TASK_STRUCT

```
struct task_struct {
#ifdef CONFIG_THREAD_INFO_IN_TASK
        struct thread_info              thread_info;
#endif
        unsigned int                    __state;
        unsigned int                    saved_state;
        randomized_struct_fields_start
        void                            *stack;
        refcount_t                      usage;
        unsigned int                    flags;
        unsigned int                    ptrace;
#ifdef CONFIG_MEM_ALLOC_PROFILING
        struct alloc_tag                *alloc_tag;
#endif

#ifdef CONFIG_SMP
        int                             on_cpu;
        struct __call_single_node       wake_entry;
        unsigned int                    wakee_flips;
        unsigned long                   wakee_flip_decay_ts;
        struct task_struct              *last_wakee;
        int                             recent_used_cpu;
        int                             wake_cpu;
#endif
        int                             on_rq;
        int                             prio;
        int                             static_prio;
        int                             normal_prio;
        unsigned int                    rt_priority;
        struct sched_entity             se;
        struct sched_rt_entity          rt;
        struct sched_dl_entity          dl;
        struct sched_dl_entity          *dl_server;
        const struct sched_class        *sched_class;
#ifdef CONFIG_SCHED_CORE
        struct rb_node                  core_node;
        unsigned long                   core_cookie;
        unsigned int                    core_occupation;
#endif
```

```
#ifdef CONFIG_CGROUP_SCHED
        struct task_group               *sched_task_group;
#endif
#ifdef CONFIG_UCLAMP_TASK
        struct uclamp_se                uclamp_req[UCLAMP_CNT];
        struct uclamp_se                uclamp[UCLAMP_CNT];
#endif
        struct sched_statistics         stats;
#ifdef CONFIG_PREEMPT_NOTIFIERS
        struct hlist_head               preempt_notifiers;
#endif
#ifdef CONFIG_BLK_DEV_IO_TRACE
        unsigned int                    btrace_seq;
#endif
        unsigned int                    policy;
        unsigned long                   max_allowed_capacity;
        int                             nr_cpus_allowed;
        const cpumask_t                 *cpus_ptr;
        cpumask_t                       *user_cpus_ptr;
        cpumask_t                       cpus_mask;
        void                            *migration_pending;
#ifdef CONFIG_SMP
        unsigned short                  migration_disabled;
#endif
        unsigned short                  migration_flags;
#ifdef CONFIG_PREEMPT_RCU
        int                             rcu_read_lock_nesting;
        union rcu_special               rcu_read_unlock_special;
        struct list_head                rcu_node_entry;
        struct rcu_node                 *rcu_blocked_node;
#endif /* #ifdef CONFIG_PREEMPT_RCU */
#ifdef CONFIG_TASKS_RCU
        unsigned long                   rcu_tasks_nvcsw;
        u8                              rcu_tasks_holdout;
        u8                              rcu_tasks_idx;
        int                             rcu_tasks_idle_cpu;
        struct list_head                rcu_tasks_holdout_list;
        int                             rcu_tasks_exit_cpu;
        struct list_head                rcu_tasks_exit_list;
#endif /* #ifdef CONFIG_TASKS_RCU */
#ifdef CONFIG_TASKS_TRACE_RCU
        int                             trc_reader_nesting;
        int                             trc_ipi_to_cpu;
        union rcu_special               trc_reader_special;
        struct list_head                trc_holdout_list;
        struct list_head                trc_blkd_node;
        int                             trc_blkd_cpu;
#endif /* #ifdef CONFIG_TASKS_TRACE_RCU */
        struct sched_info               sched_info;
        struct list_head                tasks;
#ifdef CONFIG_SMP
        struct plist_node               pushable_tasks;
        struct rb_node                  pushable_dl_tasks;
#endif
        struct mm_struct                *mm;
        struct mm_struct                *active_mm;
        struct address_space            *faults_disabled_mapping;
        int                             exit_state;
        int                             exit_code;
        int                             exit_signal;
        int                             pdeath_signal;
        unsigned long                   jobctl;
        unsigned int                    personality;
        unsigned                        sched_reset_on_fork:1;
        unsigned                        sched_contributes_to_load:1;
        unsigned                        sched_migrated:1;
        unsigned                        :0;
        unsigned                        sched_remote_wakeup:1;
#ifdef CONFIG_RT_MUTEXES
```

```
        unsigned                        sched_rt_mutex:1;
#endif
        unsigned                        in_execve:1;
        unsigned                        in_iowait:1;
#ifndef TIF_RESTORE_SIGMASK
        unsigned                        restore_sigmask:1;
#endif
#ifdef CONFIG_MEMCG
        unsigned                        in_user_fault:1;
#endif
#ifdef CONFIG_LRU_GEN
        unsigned                        in_lru_fault:1;
#endif
#ifdef CONFIG_COMPAT_BRK
        unsigned                        brk_randomized:1;
#endif
#ifdef CONFIG_CGROUPS
        unsigned                        no_cgroup_migration:1;
        unsigned                        frozen:1;
#endif
#ifdef CONFIG_BLK_CGROUP
        unsigned                        use_memdelay:1;
#endif
#ifdef CONFIG_PSI
        unsigned                        in_memstall:1;
#endif
#ifdef CONFIG_PAGE_OWNER
        unsigned                        in_page_owner:1;
#endif
#ifdef CONFIG_EVENTFD
        unsigned                        in_eventfd:1;
#endif
#ifdef CONFIG_ARCH_HAS_CPU_PASID
        unsigned                        pasid_activated:1;
#endif
#ifdef  CONFIG_CPU_SUP_INTEL
        unsigned                        reported_split_lock:1;
#endif
#ifdef CONFIG_TASK_DELAY_ACCT
        unsigned                        in_thrashing:1;
#endif
        unsigned long                   atomic_flags;
        struct restart_block            restart_block;
        pid_t                           pid;
        pid_t                           tgid;
#ifdef CONFIG_STACKPROTECTOR
        unsigned long                   stack_canary;
#endif
        struct task_struct __rcu        *real_parent;
        struct task_struct __rcu        *parent;
        struct list_head                children;
        struct list_head                sibling;
        struct task_struct              *group_leader;
        struct list_head                ptraced;
        struct list_head                ptrace_entry;
        struct pid                      *thread_pid;
        struct hlist_node               pid_links[PIDTYPE_MAX];
        struct list_head                thread_node;
        struct completion               *vfork_done;
        int __user                      *set_child_tid;
        int __user                      *clear_child_tid;
        void                            *worker_private;
        u64                             utime;
        u64                             stime;
#ifdef CONFIG_ARCH_HAS_SCALED_CPUTIME
        u64                             utimescaled;
        u64                             stimescaled;
#endif
        u64                             gtime;
```

```
        struct prev_cputime             prev_cputime;
#ifdef CONFIG_VIRT_CPU_ACCOUNTING_GEN
        struct vtime                    vtime;
#endif

#ifdef CONFIG_NO_HZ_FULL
        atomic_t                        tick_dep_mask;
#endif
        unsigned long                   nvcsw;
        unsigned long                   nivcsw;
        u64                             start_time;
        u64                             start_boottime;
        unsigned long                   min_flt;
        unsigned long                   maj_flt;
        struct posix_cputimers          posix_cputimers;
#ifdef CONFIG_POSIX_CPU_TIMERS_TASK_WORK
        struct posix_cputimers_work     posix_cputimers_work;
#endif
        const struct cred __rcu         *ptracer_cred;
        const struct cred __rcu         *real_cred;
        const struct cred __rcu         *cred;
#ifdef CONFIG_KEYS
        struct key                      *cached_requested_key;
#endif
        char                            comm[TASK_COMM_LEN];
        struct nameidata                *nameidata;
#ifdef CONFIG_SYSVIPC
        struct sysv_sem                 sysvsem;
        struct sysv_shm                 sysvshm;
#endif
#ifdef CONFIG_DETECT_HUNG_TASK
        unsigned long                   last_switch_count;
        unsigned long                   last_switch_time;
#endif
        struct fs_struct                *fs;
        struct files_struct             *files;
#ifdef CONFIG_IO_URING
        struct io_uring_task            *io_uring;
#endif
        struct nsproxy                  *nsproxy;
        struct signal_struct            *signal;
        struct sighand_struct __rcu             *sighand;
        sigset_t                        blocked;
        sigset_t                        real_blocked;
        sigset_t                        saved_sigmask;
        struct sigpending               pending;
        unsigned long                   sas_ss_sp;
        size_t                          sas_ss_size;
        unsigned int                    sas_ss_flags;
        struct callback_head            *task_works;
#ifdef CONFIG_AUDIT
#ifdef CONFIG_AUDITSYSCALL
        struct audit_context            *audit_context;
#endif
        kuid_t                          loginuid;
        unsigned int                    sessionid;
#endif
        struct seccomp                  seccomp;
        struct syscall_user_dispatch    syscall_dispatch;
        u64                             parent_exec_id;
        u64                             self_exec_id;
        spinlock_t                      alloc_lock;
        raw_spinlock_t                  pi_lock;
        struct wake_q_node              wake_q;
#ifdef CONFIG_RT_MUTEXES
        struct rb_root_cached           pi_waiters;
        struct task_struct              *pi_top_task;
        struct rt_mutex_waiter          *pi_blocked_on;
#endif
```

```
#ifdef CONFIG_DEBUG_MUTEXES
        struct mutex_waiter             *blocked_on;
#endif
#ifdef CONFIG_DEBUG_ATOMIC_SLEEP
        int                             non_block_count;
#endif
#ifdef CONFIG_TRACE_IRQFLAGS
        struct irqtrace_events          irqtrace;
        unsigned int                    hardirq_threaded;
        u64                             hardirq_chain_key;
        int                             softirqs_enabled;
        int                             softirq_context;
        int                             irq_config;
#endif
#ifdef CONFIG_PREEMPT_RT
        int                             softirq_disable_cnt;
#endif
#ifdef CONFIG_LOCKDEP
# define MAX_LOCK_DEPTH                 48UL
        u64                             curr_chain_key;
        int                             lockdep_depth;
        unsigned int                    lockdep_recursion;
        struct held_lock                held_locks[MAX_LOCK_DEPTH];
#endif
#if defined(CONFIG_UBSAN) && !defined(CONFIG_UBSAN_TRAP)
        unsigned int                    in_ubsan;
#endif
        void                            *journal_info;
        struct bio_list                 *bio_list;
        struct blk_plug                 *plug;
        struct reclaim_state            *reclaim_state;
        struct io_context               *io_context;
#ifdef CONFIG_COMPACTION
        struct capture_control          *capture_control;
#endif
        unsigned long                   ptrace_message;
        kernel_siginfo_t                *last_siginfo;
        struct task_io_accounting       ioac;
#ifdef CONFIG_PSI
        unsigned int                    psi_flags;
#endif
#ifdef CONFIG_TASK_XACCT
        u64                             acct_rss_mem1;
        u64                             acct_vm_mem1;
        u64                             acct_timexpd;
#endif
#ifdef CONFIG_CPUSETS
        nodemask_t                      mems_allowed;
        seqcount_spinlock_t             mems_allowed_seq;
        int                             cpuset_mem_spread_rotor;
        int                             cpuset_slab_spread_rotor;
#endif
#ifdef CONFIG_CGROUPS
        struct css_set __rcu            *cgroups;
        struct list_head                cg_list;
#endif
#ifdef CONFIG_X86_CPU_RESCTRL
        u32                             closid;
        u32                             rmid;
#endif
#ifdef CONFIG_FUTEX
        struct robust_list_head __user  *robust_list;
#ifdef CONFIG_COMPAT
        struct compat_robust_list_head __user *compat_robust_list;
#endif
        struct list_head                pi_state_list;
        struct futex_pi_state           *pi_state_cache;
        struct mutex                    futex_exit_mutex;
        unsigned int                    futex_state;
```

```
#endif
#ifdef CONFIG_PERF_EVENTS
        struct perf_event_context       *perf_event_ctxp;
        struct mutex                    perf_event_mutex;
        struct list_head                perf_event_list;
#endif
#ifdef CONFIG_DEBUG_PREEMPT
        unsigned long                   preempt_disable_ip;
#endif
#ifdef CONFIG_NUMA
        struct mempolicy                *mempolicy;
        short                           il_prev;
        u8                              il_weight;
        short                           pref_node_fork;
#endif
#ifdef CONFIG_NUMA_BALANCING
        int                             numa_scan_seq;
        unsigned int                    numa_scan_period;
        unsigned int                    numa_scan_period_max;
        int                             numa_preferred_nid;
        unsigned long                   numa_migrate_retry;
        u64                             node_stamp;
        u64                             last_task_numa_placement;
        u64                             last_sum_exec_runtime;
        struct callback_head            numa_work;
        struct numa_group __rcu         *numa_group;
        unsigned long                   *numa_faults;
        unsigned long                   total_numa_faults;
        unsigned long                   numa_faults_locality[3];
        unsigned long                   numa_pages_migrated;
#endif /* CONFIG_NUMA_BALANCING */
#ifdef CONFIG_RSEQ
        struct rseq __user *rseq;
        u32 rseq_len;
        u32 rseq_sig;
        unsigned long rseq_event_mask;
#endif
#ifdef CONFIG_SCHED_MM_CID
  int                           mm_cid;         /* Current cid in mm */
  int                           last_mm_cid;    /* Most recent cid in mm */
  int                           migrate_from_cpu;
  int                           mm_cid_active;  /* Whether cid bitmap is active */
        struct callback_head            cid_work;
#endif
        struct tlbflush_unmap_batch     tlb_ubc;
        struct pipe_inode_info          *splice_pipe;
        struct page_frag                task_frag;
#ifdef CONFIG_TASK_DELAY_ACCT
        struct task_delay_info          *delays;
#endif
#ifdef CONFIG_FAULT_INJECTION
        int                             make_it_fail;
        unsigned int                    fail_nth;
#endif
        int                             nr_dirtied;
        int                             nr_dirtied_pause;
        unsigned long                   dirty_paused_when;

#ifdef CONFIG_LATENCYTOP
        int                             latency_record_count;
        struct latency_record           latency_record[LT_SAVECOUNT];
#endif
        u64                             timer_slack_ns;
        u64                             default_timer_slack_ns;
#if defined(CONFIG_KASAN_GENERIC) || defined(CONFIG_KASAN_SW_TAGS)
        unsigned int                    kasan_depth;
#endif
#ifdef CONFIG_KCSAN
        struct kcsan_ctx                kcsan_ctx;
```

```
#ifdef CONFIG_TRACE_IRQFLAGS
        struct irqtrace_events          kcsan_save_irqtrace;
#endif
#ifdef CONFIG_KCSAN_WEAK_MEMORY
        int                             kcsan_stack_depth;
#endif
#endif
#ifdef CONFIG_KMSAN
        struct kmsan_ctx                kmsan_ctx;
#endif
#if IS_ENABLED(CONFIG_KUNIT)
        struct kunit                    *kunit_test;
#endif
#ifdef CONFIG_FUNCTION_GRAPH_TRACER
        int                             curr_ret_stack;
        int                             curr_ret_depth;
        struct ftrace_ret_stack         *ret_stack;
        unsigned long long              ftrace_timestamp;
        atomic_t                        trace_overrun;
        atomic_t                        tracing_graph_pause;
#endif
#ifdef CONFIG_TRACING
        unsigned long                   trace_recursion;
#endif /* CONFIG_TRACING */
#ifdef CONFIG_KCOV
        unsigned int                    kcov_mode;
        unsigned int                    kcov_size;
        void                            *kcov_area;
        struct kcov                     *kcov;
        u64                             kcov_handle;
        int                             kcov_sequence;
        unsigned int                    kcov_softirq;
#endif
#ifdef CONFIG_MEMCG
        struct mem_cgroup               *memcg_in_oom;
        unsigned int                    memcg_nr_pages_over_high;
        struct mem_cgroup               *active_memcg;
#endif
#ifdef CONFIG_MEMCG_KMEM
        struct obj_cgroup               *objcg;
#endif
#ifdef CONFIG_BLK_CGROUP
        struct gendisk                  *throttle_disk;
#endif
#ifdef CONFIG_UPROBES
        struct uprobe_task              *utask;
#endif
#if defined(CONFIG_BCACHE) || defined(CONFIG_BCACHE_MODULE)
        unsigned int                    sequential_io;
        unsigned int                    sequential_io_avg;
#endif
        struct kmap_ctrl                kmap_ctrl;
#ifdef CONFIG_DEBUG_ATOMIC_SLEEP
        unsigned long                   task_state_change;
# ifdef CONFIG_PREEMPT_RT
        unsigned long                   saved_state_change;
# endif
#endif
        struct rcu_head                 rcu;
        refcount_t                      rcu_users;
        int                             pagefault_disabled;
#ifdef CONFIG_MMU
        struct task_struct              *oom_reaper_list;
        struct timer_list               oom_reaper_timer;
#endif
#ifdef CONFIG_VMAP_STACK
        struct vm_struct                *stack_vm_area;
#endif
#ifdef CONFIG_THREAD_INFO_IN_TASK
```

```
        refcount_t                      stack_refcount;
#endif
#ifdef CONFIG_LIVEPATCH
        int patch_state;
#endif
#ifdef CONFIG_SECURITY
        void                            *security;
#endif
#ifdef CONFIG_BPF_SYSCALL
        struct bpf_local_storage __rcu  *bpf_storage;
        struct bpf_run_ctx              *bpf_ctx;
#endif
#ifdef CONFIG_GCC_PLUGIN_STACKLEAK
        unsigned long                   lowest_stack;
        unsigned long                   prev_lowest_stack;
#endif
#ifdef CONFIG_X86_MCE
        void __user                     *mce_vaddr;
        __u64                           mce_kflags;
        u64                             mce_addr;
        __u64                           mce_ripv : 1,
                                        mce_whole_page : 1,
                                        __mce_reserved : 62;
        struct callback_head            mce_kill_me;
        int                             mce_count;
#endif
#ifdef CONFIG_KRETPROBES
        struct llist_head               kretprobe_instances;
#endif
#ifdef CONFIG_RETHOOK
        struct llist_head               rethooks;
#endif
#ifdef CONFIG_ARCH_HAS_PARANOID_L1D_FLUSH
        struct callback_head            l1d_flush_kill;
#endif
#ifdef CONFIG_RV
        union rv_task_monitor           rv[RV_PER_TASK_MONITORS];
#endif

#ifdef CONFIG_USER_EVENTS
        struct user_event_mm            *user_event_mm;
#endif
        randomized_struct_fields_end
        struct thread_struct            thread;
};
```

```
/* Used in tsk->__state: */
#define TASK_RUNNING                    0x00000000
#define TASK_INTERRUPTIBLE              0x00000001
#define TASK_UNINTERRUPTIBLE            0x00000002
#define __TASK_STOPPED                  0x00000004
#define __TASK_TRACED                   0x00000008
/* Used in tsk->exit_state: */
#define EXIT_DEAD                       0x00000010
#define EXIT_ZOMBIE                     0x00000020
#define EXIT_TRACE                      (EXIT_ZOMBIE | EXIT_DEAD)
/* Used in tsk->__state again: */
#define TASK_PARKED                     0x00000040
#define TASK_DEAD                       0x00000080
#define TASK_WAKEKILL                   0x00000100
#define TASK_WAKING                     0x00000200
#define TASK_NOLOAD                     0x00000400
#define TASK_NEW                        0x00000800
#define TASK_RTLOCK_WAIT                0x00001000
#define TASK_FREEZABLE                  0x00002000
#define __TASK_FREEZABLE_UNSAFE        (0x00004000 * IS_ENABLED(CONFIG_LOCKDEP))
#define TASK_FROZEN                     0x00008000
#define TASK_STATE_MAX                  0x00010000
```

```
#define PF_VCPU                 0x00000001      /* I'm a virtual CPU */
#define PF_IDLE                 0x00000002      /* I am an IDLE thread */
#define PF_EXITING              0x00000004      /* Getting shut down */
#define PF_POSTCOREDUMP         0x00000008      /* Coredumps should ignore this task */
#define PF_IO_WORKER            0x00000010      /* Task is an IO worker */
#define PF_WQ_WORKER            0x00000020      /* I'm a workqueue worker */
#define PF_FORKNOEXEC           0x00000040      /* Forked but didn't exec */
#define PF_MCE_PROCESS          0x00000080      /* Process policy on mce errors */
#define PF_SUPERPRIV            0x00000100      /* Used super-user privileges */
#define PF_DUMPCORE             0x00000200      /* Dumped core */
#define PF_SIGNALED             0x00000400      /* Killed by a signal */
#define PF_MEMALLOC             0x00000800      /* Allocating memory to free memory. See
memalloc_noreclaim_save() */
#define PF_NPROC_EXCEEDED       0x00001000      /* set_user() noticed that RLIMIT_NPROC
was exceeded */
#define PF_USED_MATH            0x00002000      /* If unset the fpu must be initialized
before use */
#define PF_USER_WORKER          0x00004000      /* Kernel thread cloned from userspace
thread */
#define PF_NOFREEZE             0x00008000      /* This thread should not be frozen */
#define PF__HOLE__00010000      0x00010000
#define PF_KSWAPD               0x00020000      /* I am kswapd */
#define PF_MEMALLOC_NOFS        0x00040000      /* All allocations inherit GFP_NOFS. See
memalloc_nfs_save() */
#define PF_MEMALLOC_NOIO        0x00080000      /* All allocations inherit GFP_NOIO. See
memalloc_noio_save() */
#define PF_LOCAL_THROTTLE       0x00100000      /* Throttle writes only against the bdi I
write to,                                * I am cleaning dirty pages from some other bdi. */
#define PF_KTHREAD              0x00200000      /* I am a kernel thread */
#define PF_RANDOMIZE            0x00400000      /* Randomize virtual address space */
#define PF_MEMALLOC_NORECLAIM   0x00800000      /* All allocation requests will clear
__GFP_DIRECT_RECLAIM */
#define PF_MEMALLOC_NOWARN      0x01000000      /* All allocation requests will inherit
__GFP_NOWARN */
#define PF__HOLE__02000000      0x02000000
#define PF_NO_SETAFFINITY       0x04000000      /* Userland is not allowed to meddle with
cpus_mask */
#define PF_MCE_EARLY            0x08000000      /* Early kill for mce process policy */
#define PF_MEMALLOC_PIN         0x10000000      /* Allocations constrained to zones which
allow long term pinning.                 * See memalloc_pin_save() */
#define PF_BLOCK_TS             0x20000000      /* plug has ts that needs updating */
#define PF__HOLE__40000000      0x40000000
#define PF_SUSPEND_TASK         0x80000000      /* This thread called freeze_processes()
and should not be frozen */
```

```
struct stat {
  unsigned long st_dev ; /* Device . */
  unsigned long st_ino ; /* File serial number . */
  unsigned int st_mode ; /* File mode . */
  unsigned int st_nlink ; /* Link count . */
  unsigned int st_uid ; /* User ID of the file 's owner . */
  unsigned int st_gid ; /* Group ID of the file 's group . */
  unsigned long st_rdev ; /* Device number , if device . */
  unsigned long __pad1 ;
  long st_size ; /* Size of file , in bytes . */
  int st_blksize ; /* Optimal block size for I / O . */
  int __pad2 ;
  long st_blocks ; /* Number 512 - byte blocks allocated . */
  long st_atime ; /* Time of last access . */
  unsigned long st_atime_nsec ;
  long st_mtime ; /* Time of last modification . */
  unsigned long st_mtime_nsec ;
  long st_ctime ; /* Time of last status change . */
  unsigned long st_ctime_nsec ;
  unsigned int __unused4 ;
  unsigned int __unused5 ;
};
```

```
typedef struct _IO_FILE FILE;

struct _IO_FILE
{
  int _flags;           /* High-order word is _IO_MAGIC; rest is flags. */
  /* The following pointers correspond to the C++ streambuf protocol. */
  char *_IO_read_ptr;   /* Current read pointer */
  char *_IO_read_end;   /* End of get area. */
  char *_IO_read_base;  /* Start of putback+get area. */
  char *_IO_write_base; /* Start of put area. */
  char *_IO_write_ptr;  /* Current put pointer. */
  char *_IO_write_end;  /* End of put area. */
  char *_IO_buf_base;   /* Start of reserve area. */
  char *_IO_buf_end;    /* End of reserve area. */
  /* The following fields are used to support backing up and undo. */
  char *_IO_save_base; /* Pointer to start of non-current get area. */
  char *_IO_backup_base;  /* Pointer to first valid character of backup area */
  char *_IO_save_end; /* Pointer to end of non-current get area. */
  struct _IO_marker *_markers;
  struct _IO_FILE *_chain;
  int _fileno;
  int _flags2;
  __off_t _old_offset; /* This used to be _offset but it's too small.  */
  /* 1+column number of pbase(); 0 is unknown. */
  unsigned short _cur_column;
  signed char _vtable_offset;
  char _shortbuf[1];

  _IO_lock_t *_lock;
#ifdef _IO_USE_OLD_IO_FILE
};
```

```
struct urb {
        /* private: usb core and host controller only fields in the urb */
        struct kref kref;               /* reference count of the URB */
        int unlinked;                   /* unlink error code */
        void *hcpriv;                   /* private data for host controller */
        atomic_t use_count;             /* concurrent submissions counter */
        atomic_t reject;                /* submissions will fail */
        /* public: documented fields in the urb that can be used by drivers */
        struct list_head urb_list;      /* list head for use by the urb's
                                         * current owner */
        struct list_head anchor_list;   /* the URB may be anchored */
        struct usb_anchor *anchor;
```

```
        struct usb_device *dev;         /* (in) pointer to associated device */
        struct usb_host_endpoint *ep;   /* (internal) pointer to endpoint */
        unsigned int pipe;              /* (in) pipe information */
        unsigned int stream_id;         /* (in) stream ID */
        int status;                     /* (return) non-ISO status */
        unsigned int transfer_flags;    /* (in) URB_SHORT_NOT_OK | ...*/
        void *transfer_buffer;          /* (in) associated data buffer */
        dma_addr_t transfer_dma;        /* (in) dma addr for transfer_buffer */
        struct scatterlist *sg;         /* (in) scatter gather buffer list */
        int num_mapped_sgs;             /* (internal) mapped sg entries */
        int num_sgs;                    /* (in) number of entries in the sg list */
        u32 transfer_buffer_length;     /* (in) data buffer length */
        u32 actual_length;              /* (return) actual transfer length */
        unsigned char *setup_packet;    /* (in) setup packet (control only) */
        dma_addr_t setup_dma;           /* (in) dma addr for setup_packet */
        int start_frame;                /* (modify) start frame (ISO) */
        int number_of_packets;          /* (in) number of ISO packets */
        int interval;                   /* (modify) transfer interval
                                         * (INT/ISO) */
        int error_count;                /* (return) number of ISO errors */
        void *context;                  /* (in) context for completion */
        usb_complete_t complete;        /* (in) completion routine */
        struct usb_iso_packet_descriptor iso_frame_desc[];
                                        /* (in) ISO ONLY */
};
```

```
struct usb_interface {
        /* array of alternate settings for this interface,
         * stored in no particular order */
        struct usb_host_interface *altsetting;

        struct usb_host_interface *cur_altsetting;      /* the currently
                                         * active alternate setting */
        unsigned num_altsetting;        /* number of alternate settings */

        /* If there is an interface association descriptor then it will list
         * the associated interfaces */
        struct usb_interface_assoc_descriptor *intf_assoc;

        int minor;                      /* minor number this interface is
                                         * bound to */
        enum usb_interface_condition condition;         /* state of binding */
        unsigned sysfs_files_created:1; /* the sysfs attributes exist */
        unsigned ep_devs_created:1;     /* endpoint "devices" exist */
        unsigned unregistering:1;       /* unregistration is in progress */
        unsigned needs_remote_wakeup:1; /* driver requires remote wakeup */
        unsigned needs_altsetting0:1;   /* switch to altsetting 0 is pending */
        unsigned needs_binding:1;       /* needs delayed unbind/rebind */
        unsigned resetting_device:1;    /* true: bandwidth alloc after reset */
        unsigned authorized:1;          /* used for interface authorization */
        enum usb_wireless_status wireless_status;
        struct work_struct wireless_status_work;

        struct device dev;              /* interface specific device info */
        struct device *usb_dev;
        struct work_struct reset_ws;    /* for resets in atomic context */
};
```