The struct taskstats

This document contains an explanation of the struct taskstats fields.

There are three different groups of fields in the struct taskstats:

1. Common and basic accounting fields

   If CONFIG_TASKSTATS is set, the taskstats interface is enabled and the common fields and basic accounting fields are collected for delivery at do_exit() of a task.

2. Delay accounting fields

   These fields are placed between:

   ```
   /* Delay accounting fields start */
   ```

   and:

   ```
   /* Delay accounting fields end */
   ```

   Their values are collected if CONFIG_TASK_DELAY_ACCT is set.

3. Extended accounting fields

   These fields are placed between:

   ```
   /* Extended accounting fields start */
   ```

   and:

   ```
   /* Extended accounting fields end */
   ```

   Their values are collected if CONFIG_TASK_XACCT is set.

4. Per-task and per-thread context switch count statistics
5. Time accounting for SMT machines
6. Extended delay accounting fields for memory reclaim

Future extension should add fields to the end of the taskstats struct, and should not change the relative position of each field within the struct.

```
struct taskstats {
```

1. Common and basic accounting fields:

```
/* The version number of this struct. This field is always set to
 * TAKSTATS_VERSION, which is defined in <linux/taskstats.h>.
 * Each time the struct is changed, the value should be incremented.
 */
__u16   version;

/* The exit code of a task. */
__u32   ac_exitcode;            /* Exit status */

/* The accounting flags of a task as defined in <linux/acct.h>
 * Defined values are AFORK, ASU, ACOMPAT, ACORE, and AXSIG.
 */
__u8    ac_flag;                /* Record flags */

/* The value of task_nice() of a task. */
__u8    ac_nice;                /* task_nice */

/* The name of the command that started this task. */
char    ac_comm[TS_COMM_LEN];   /* Command name */

/* The scheduling discipline as set in task->policy field. */
__u8    ac_sched;               /* Scheduling discipline */

__u8    ac_pad[3];
__u32   ac_uid;                 /* User ID */
__u32   ac_gid;                 /* Group ID */
__u32   ac_pid;                 /* Process ID */
__u32   ac_ppid;                /* Parent process ID */

/* The time when a task begins, in [secs] since 1970. */
__u32   ac_btime;               /* Begin time [sec since 1970] */

/* The elapsed time of a task, in [usec]. */
__u64   ac_etime;               /* Elapsed time [usec] */
```

```
/* The user CPU time of a task, in [usec]. */
__u64   ac_utime;               /* User CPU time [usec] */

/* The system CPU time of a task, in [usec]. */
__u64   ac_stime;               /* System CPU time [usec] */

/* The minor page fault count of a task, as set in task->min_flt. */
__u64   ac_minflt;              /* Minor Page Fault Count */

/* The major page fault count of a task, as set in task->maj_flt. */
__u64   ac_majflt;              /* Major Page Fault Count */
```

2. Delay accounting fields:

```
/* Delay accounting fields start
 *
 * All values, until the comment "Delay accounting fields end" are
 * available only if delay accounting is enabled, even though the last
 * few fields are not delays
 *
 * xxx_count is the number of delay values recorded
 * xxx_delay_total is the corresponding cumulative delay in nanoseconds
 *
 * xxx_delay_total wraps around to zero on overflow
 * xxx_count incremented regardless of overflow
 */

/* Delay waiting for cpu, while runnable
 * count, delay_total NOT updated atomically
 */
__u64   cpu_count;
__u64   cpu_delay_total;

/* Following four fields atomically updated using task->delays->lock */

/* Delay waiting for synchronous block I/O to complete
 * does not account for delays in I/O submission
 */
__u64   blkio_count;
__u64   blkio_delay_total;

/* Delay waiting for page fault I/O (swap in only) */
__u64   swapin_count;
__u64   swapin_delay_total;

/* cpu "wall-clock" running time
 * On some architectures, value will adjust for cpu time stolen
 * from the kernel in involuntary waits due to virtualization.
 * Value is cumulative, in nanoseconds, without a corresponding count
 * and wraps around to zero silently on overflow
 */
__u64   cpu_run_real_total;

/* cpu "virtual" running time
 * Uses time intervals seen by the kernel i.e. no adjustment
 * for kernel's involuntary waits due to virtualization.
 * Value is cumulative, in nanoseconds, without a corresponding count
 * and wraps around to zero silently on overflow
 */
__u64   cpu_run_virtual_total;
/* Delay accounting fields end */
/* version 1 ends here */
```

3. Extended accounting fields:

```
/* Extended accounting fields start */

/* Accumulated RSS usage in duration of a task, in MBytes-usecs.
 * The current rss usage is added to this counter every time
 * a tick is charged to a task's system time. So, at the end we
 * will have memory usage multiplied by system time. Thus an
 * average usage per system time unit can be calculated.
 */
__u64   coremem;                /* accumulated RSS usage in MB-usec */

/* Accumulated virtual memory usage in duration of a task.
 * Same as acct_rss_mem1 above except that we keep track of VM usage.
 */
__u64   virtmem;                /* accumulated VM usage in MB-usec */

/* High watermark of RSS usage in duration of a task, in KBytes. */
```

```
        __u64   hiwater_rss;            /* High-watermark of RSS usage */

        /* High watermark of VM  usage in duration of a task, in KBytes. */
        __u64   hiwater_vm;             /* High-water virtual memory usage */

        /* The following four fields are I/O statistics of a task. */
        __u64   read_char;              /* bytes read */
        __u64   write_char;             /* bytes written */
        __u64   read_syscalls;          /* read syscalls */
        __u64   write_syscalls;         /* write syscalls */

        /* Extended accounting fields end */
```

4. Per-task and per-thread statistics:

```
        __u64   nvcsw;                  /* Context voluntary switch counter */
        __u64   nivcsw;                 /* Context involuntary switch counter */
```

5. Time accounting for SMT machines:

```
        __u64   ac_utimescaled;         /* utime scaled on frequency etc */
        __u64   ac_stimescaled;         /* stime scaled on frequency etc */
        __u64   cpu_scaled_run_real_total; /* scaled cpu_run_real_total */
```

6. Extended delay accounting fields for memory reclaim:

```
        /* Delay waiting for memory reclaim */
        __u64   freepages_count;
        __u64   freepages_delay_total;

}
```