Implementation of virtual time in kernel

Variable added in ***sched_entity*** structure :

```
    volatile u64              on_cpu_time;
```

On_cpu_time stores the CPU execution time for self task.

```
  volatile u64              mx_on_cpu_time;
```

mx_on_cpu_time stores the maximum CPU execution time over all child task including itesef.

```
     u64                      del_exec;
     u64                      base_on_cpu_time;
     u64                      naive_vtime;
     u64                      mx_naive_vtime;
```

These variable were used for testing purpose.

```
     struct list_head child_vtime_at_exit;
     spinlock_t child_vtlist_lock;
```

Every time a child task terminates it reports its on_cpu_time to its parent by enqueuing own on_cpu_time into parent's child_vtime_at_exit list. The lock is used to ensure mutual exclusion while multiple children try to access the list concurrently.

The added variables in ***sched_entity*** structure are initialized in ***__sched_fork()*** function:

```
    p->se.on_cpu_time             = current->se.on_cpu_time;
     p->se.base_on_cpu_time         = current->se.on_cpu_time;
     p->se.mx_on_cpu_time           = 0;
     INIT_LIST_HEAD(&p->se.child_vtime_at_exit);
     p->se.naive_vtime             = 0;
     p->se.mx_naive_vtime           = 0;
     spin_lock_init(&p->se.child_vtlist_lock);
```

Updating the ***on_cpu_time*** variable in ***update_curr()*** function:

```
        curr->on_cpu_time += delta_exec;
        curr->del_exec = delta_exec;
        curr->naive_vtime += delta_exec;
```

Updating the ***mx_on_cpu_time*** variable in ***do_exit()*** function:

```
 struct task_struct *parent = tsk->group_leader;
        if (parent->se.mx_on_cpu_time <= tsk->se.on_cpu_time){
                parent->se.mx_on_cpu_time = tsk->se.on_cpu_time;
        }
        parent->se.mx_naive_vtime = parent->se.mx_naive_vtime > tsk->se.naive_vtime?
parent->se.mx_naive_vtime : tsk->se.naive_vtime;
```

Enqueuing own ***on_cpu_time*** in the parent's ***child_vtime_at_exit*** list at the task termination in ***do_exit()*** function:

```
extern int pthread_join_activated;

        if (pthread_join_activated >0 ){
                struct vtime_struct *tmp;
                struct list_head *pos, *q;
                // delete every child vtime record
                list_for_each_safe(pos, q, &current->se.child_vtime_at_exit){
                        tmp= list_entry(pos, struct vtime_struct, next);
                        //printk(KERN_INFO "freeing item pid= %d\n", tmp->pid);
                        list_del(pos);
                        kfree(tmp);
                }
```

```
		if (tsk->pid != parent->pid){

			struct vtime_struct *vtst = kmalloc(sizeof(struct
vtime_struct),GFP_KERNEL);
			if (vtst){
				vtst->pid = tsk->pid;
				vtst->vtime = tsk->se.on_cpu_time;
				//kfree(vtst);
				spin_lock(&parent->se.child_vtlist_lock);
				list_add ( &vtst->next , &parent->se.child_vtime_at_exit ) ;
				spin_unlock(&parent->se.child_vtlist_lock);

			}

		}
	}
```