KERNEL.DATA.STRUCTURES

Task Structure

The fundamental entity of any operating system is the task or a process. The task structure is defined in the **linux/sched.h** and it is the important data structure and following are some of its parameters.

struct task_struct {

volatile long state; //determines the current state of the process like unrunnable, runnable and stopped

unsigned long flags; // various flags for accounting purpose like PF_STARTIN, PF_MEMALLOC

unsigned long ptrace; // process is being monitored by other process.

int sigpending; //define when the signals must be handed over to this process

mm_segment_t addr_limit;

struct exec_domain *exec_domain; //domain other than x86

long need_resched; //flag indicates that scheduling must be executed

int lock_depth; //structure is protected before simulataneous access can take place

long counter; //dynamic priority of a process

long nice; //static priority of a process

unsigned long policy; //scheduling policy like SCHED_RR, SCHED_FIFO, SCHED_OTHER

unsigned long rt_priority;

.

};

Process Relations

This data structure describes the relation between processes lilke child process, parent process, etc.

struct task_struct *p_opptr; //original parent process

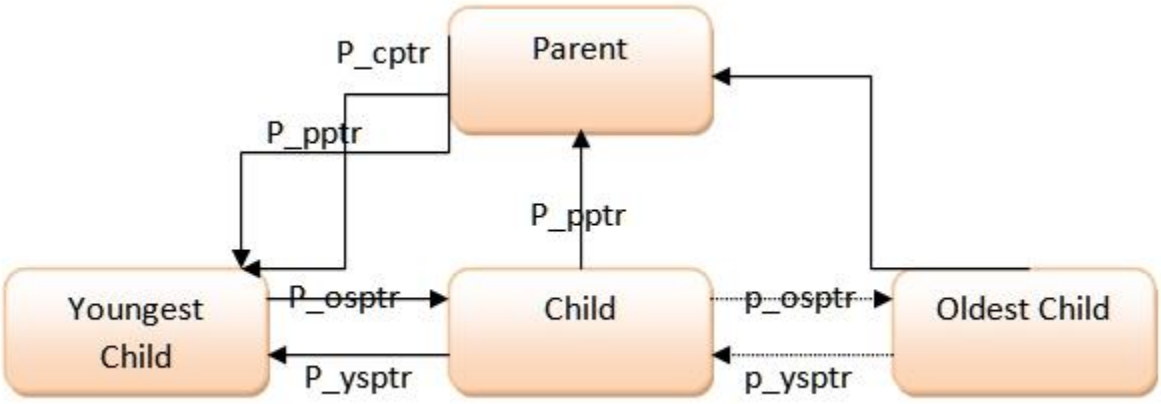
struct task_struct *p_pptr; //Parent Process

struct task_struct *p_cptr; //child process

struct task_struct *p_ysptr; //youngest sibling process

struct task_struct *p_osptr; //old sibling process

the following diagram depicts the relationship between the process.



Process ID

Every process has its own process ID called pid defined as follows:

pid_t pid, pgrp, session, tgid; //pgrp – process group, session in which the process runs and tgid is thread group id

int leader; //there will be a leader process among the process

apart from the above, there are two other parameters like **uid** and **gid** referred namely as user identification and group identification, which enables to create the access right.

Timing

Times are measured in ticks. usually the timing signals are generated by the hardware timer every10ms which is captured by the timer interrupt.

there are different timing parameters are specified like

long per_cpu_utime[NO OF CPUS]; //user mode

long per_cpu_stime[NO OF CPUS]; //system mode

struct tms times; //the timing values are added for all the child processes and other processes

unsigned long start_time; //time at which the process was generated.

FILES and INODES

Files and inodes represent the file structures.

Files defines the attributes of file like **mode of operation, position of the cursor, flags, reference count, directory entry, etc**

inodes defines the information about the files **like owner of the file, device in which the file is located, access rights of the file, size of the file, last access time, last modification time, etc**

struct file {

mode_t f_mode; // mode of operation like read, write, append, etc

loff_t f_pos; //position of the read/write cursor

atomic_t f_count; //reference count like how many times the file has been opened by the system call

unsigned int f_flags; //access control

struct dentry *fs_dentry;// file descriptor

.

};

struct inode {

kdev_t i_dev:// device id in which the file is located

unsigned long i_ino;//file within device

umode_t i_mode; //mode of operation

uid_t i_uid; //user id which is the owner of the file

gid_t i_gid; //group id, the file can be accessed by a group

off_t i_size; //size of the file in bytes

time_t i_mtime; //last modification time

time_t i_atime; //last accessed time

.

};

Source : http://engineeringcourses.files.wordpress.com/2010/04/osp-lecture-notes1.pdf