COMP304 - Operating Systems
Problem Session
Project-1

Project Expectations (Summary)

- Accept an input command
- If valid execute that command
- Following features should be supported by the myshell
  - Run the command in background (&) or foreground
  - Implement history feature
  - Implement three new commands bookmark, muzik and codesearch
  - Implement a command of your choice
  - Implement a kernel module that outputs the characteristics of a process
  - Implement that kernel module as a command processInfo in myshell

Creating a process

- In Linux a process is created by calling fork(), this will duplicate the existing one.
- process that calls fork() is the parent, whereas the new process is the child.
- The parent resumes execution and the child starts execution at the same place, where the call returns.
- Finally, a program exits via the exit() system call. This function terminates the process and frees all its resources.
- A parent process can inquire about the status of a terminated child via the waitpid() system call, which enables a process to wait for the termination of a specific process.

execv

- Replaces the current process image with a new process image
- int execv(const char *path, char *const argv[])
- Execv requires a full path to the executable and arguments as string
- You can tokenize the contents of PATH environment variable
- Check for the command in all directories included in the PATH
- You can use access() .

Running in background(&)

- Fork a child process
- In the child
  - Execute the command
  - Exit the child process
- In the parent
  - if need to run in background do not wait
  - if foreground process wait for the child process to complete execution

Linux crontab

The **crontab** is a list of commands that you want to run on a regular schedule, and also the name of the command used to manage that list.

Get the crontab:

sudo apt-get install gnome-schedule

Using crontab

$ sudo crontab –e

$ 01 04 1 1 1 /usr/bin/somedirectory/somecommand

// run /usr/bin/somedirectory/somecommand at 4:01am on January 1st plus every Monday in January.

General format of the commands:

minute (0-59), hour (0-23, 0 = midnight), day (1-31), month (1-12), weekday (0-6, 0 = Sunday)

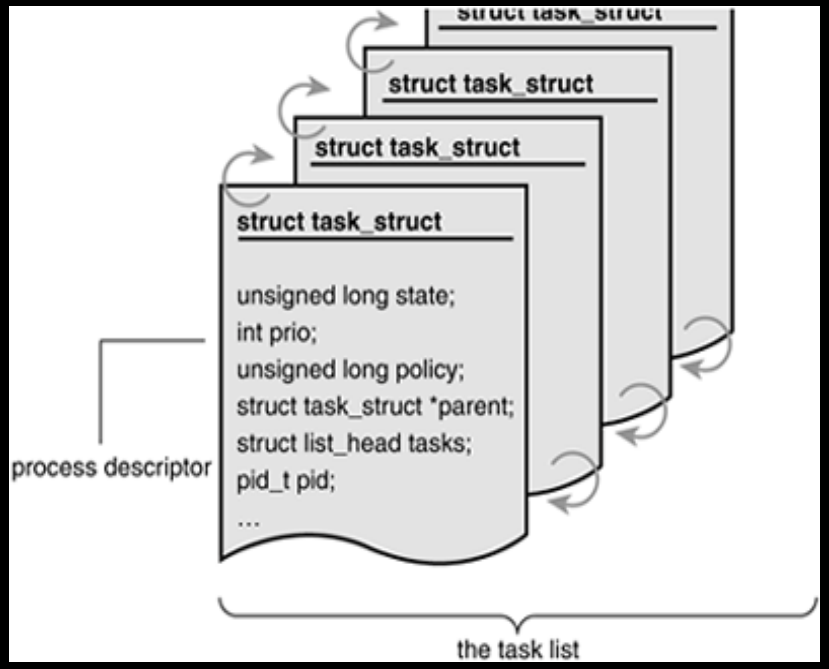
* Asterisk application

$01 04 * * * /usr/bin/somedirectory/somecommand

The above example will run /usr/bin/somedirectory/somecommand at 4:01am on every day of every month.

Linux Task Structure

- The kernel stores the list of processes in a circular doubly linked list called the task list
- Each element in the task list is a process descriptor of the type struct task_struct, which is defined in <linux/sched.h>.



Linux task_struct

- Each task_struct has a pointer to the parent's task_struct, named parent, and a list of children, named children.

  struct task_struct parent;

  struct list_head children;

- Process id:

  pid_t pid;

- Executables name excluding path

  char comm[TASK_COMM_LEN];

- Scheduling related information

  int prio, static_prio; unsigned int policy;

for_each_process

```
struct task_struct *task;


for_each_process(task)
{
   /* this prints the name and PID of each task */
   printk("%s[%d]\n", task->comm, task->pid);
}
```

list_for_each

```
struct list_head *p;


list_for_each(p, list) {
        /* p points to an entry in the list */
}
```

list_entry

list_entry(ptr,type,member): return a pointer to the structure "type" that contains "member" which is of type of “ptr”

task1=list_entry(list,struct task_struct,children);

The above call, returns the pointer to the structure "task_struct" that contains the “children".

Once the pointer to task_struct is known, we can print the process name and process id using

task1->comm and task1->pid.