Monthly Planner

January

i. Interactive process :- They instantly interacts with their users, therefore they spend lot of time with keyboard and mouse operations.
- The average dealy must be falls on 50 and 150 milliseconds.
- The variance of the average dealy of the system must be erratic.
- Typically interactive programs are command shell, text editors, Graphic -cs applications.

ii. Batch process :- They do not use user ~~interaction~~, but they often run in the background.
- Batch programs are programming lan -guage compilers, database search en -gines, scientific computations, Data -backup, software updates etc.

iii. real-time process :- This process is ~~stringent~~ static process.
- real-time process are video and au -dio and robot controllers, the pro -cess is collection of data to the structures and sca -mers etc.

A. System calls to the scheduler

Appointments - queue list. process is a fairness ass.
- gonment of cpu time all sched - RR used
- in a same priority of a process.

iii. sched_normal :- it is a conventional process.
- Scheduled algorithm is a diffently depe
-nd on the behaviour of the convention
-al process.
- Every conventional process is having ow
-n its static priority.
- The kernal of the scheduled algorithm of the number ranging from 100 to 139.



- The new process is inherits from its parent.
- process is changes from the System calls nice() and setpriori
-ty().
- The real-time process is having a real-time priority(). value changing from 1to99.
- The real-time priority changes from the set priority and sched_set_scheduler() System calls.

☆. Basic time quantum

- The static priority is essentially deter
-mines the basic time quantum pro
-cess. quantum time duration is assigned to the process it has exhaust
-ed previous time of quantum.

6 Monday JANUARY 2020

- Static priority and base time quantum are related by the following formula:-

$$\text{base time quantum (in milliseconds)} = \begin{cases} (140 - \text{Static priority}) \times 20 & \text{if Static priority} < 120 \\ (140 - \text{Static priority}) \times 5 & \text{if Static priority} \geq 120 \end{cases}$$

## 2. Dynamic priority

- Static priority, a conventional process also dynamic priority which value ranging from 100 to 139.
- It is related to the static priority by following empirical formula

dynamic priority = max(100, min(static priority - bonus +5, 139))

## 3. Data structure used by the scheduler

- runqueue is most important data structure for a Linux 2.6 scheduler.
- Each of the cpu system having own

9 Thursday JANUARY 2020

Week 2 ○ (009-357)

Appointments

# I. Process in Linux

## 1. Process ~~Descriptor~~ Lightweight processes and threads

- The term "process" often used with several different meanings. and process is a instance of program in execution.
- processes are like human beings: they generated, they have a more or less significant life, they optionally generate one or more child process, and eventually they die.
- From the kernal point of view, the purpose of the process is act as an entity to which system resource (cpu, time, memory) etc. are allocated.
- when a process is created, it is almost identical to its parent.
- It receives the copy of the parent address space and executes the same code as parent, beginning at the next instruction following the process creation system call.
- it supports multithreaded application user programs having many relatively independent Execution flows sharing large portion of the application datastructures.
- In most of the multithreaded applic

"Do one thing every day that scares you." - Eleanor Roosevelt

10

-ations are written using standard set of library functions called pthread (POSIX thread) libraries.
• older version of the linux kernel offered no support for a multithreaded applicati -ons. From kernel point of view, mul -tithreaded application was a just normal process.
• The multiple execution flows of a multi -threaded application was created, hand -led and scheduled entirely user mode.
• For instance, suppose a chess program uses two threads: one of them control a Graphical chessboards, waiting for the moves from the human player and showi -ng the moves of the computers, while ot -her thread ponders the next move of the game.
• While first thread wait for the human move, the second thread should run continuously thus exploiting thinkin -g of the human player.
• A straight forward way to impleme -nt multithreaded applications is to associate a lightweight process with each thread.
• Examples of the POSIX-compliant pth -read libraries that use light-weight process are Linux threads, native POSIX Thread library, IBM next Generation posix threading package.

Appointments

• Posix compliant multithreaded applications are best handled by kernels that support "thread groups".
• In Linux a thread group is basically a set of lightweight processes that implement a multithreaded application and act as whole with regards to some system calls such as getpid(), kill() and exit().

2. Process Switch

i. process switching :- In order to control the execution of the processes, the kernel must be able to suspended processes running on the cpu. and resume execution of the processes previously suspended is called a process switching, task switching and context switching.



Zanzibar Revolution Day (Tanzania)

ii. Hardware context :- • while each process can have its own address space, all processes have to share the cpu registers. So before resuming the execution of the process, the kernel must ensure each registers is loaded with the value.

• Set of data must be loaded in to the registers before the process resumes execution on the cpu is called a

-hardware context.
• we will assume the prev local variable refers to the process descriptor of the process being switched out and next refers to the one being switched in to replace it.
• we can define process switch as the activity consisting of saving the hardware context prev. and replacing with the hardware context of next.
• older version of the linux took advantage of the hardware support offered by the 80x86 architecture and performed a process switch through a far jmp instructions to the selector of the Task state segment descriptor to the next process.
• Linux 2.6 software to perform a process switch for the following process:-

i. For jmp instructions modify both the CS and EIP registers, while simple jmp instructions modify only EIP.
ii. Step-by-Step switching performed through a sequence of mov instructions allows better to control over the validity of data being loaded.
iii. The amount of time required old approach and new approach become same.
iv. process switching occurs only in the kernel mode.

iii. Task State Segment :-

Appointments

v. Switch_to() function

- Switching to new task :- we can switching from one task to another, we call Switch_to() function.
- Saving Registers :- "Switch_to starts by saving the registers (like eax, edx etc) of the current task (Task A). it's like taking the snapshot of everything the task is currently hold.
- Updating the task :- now update the task Switch_to (TaskB). it load the TaskB registers values. it's like TaskB starting with all right information.
- Restoring Registers :- when we switch back to A later, "Switch_to()" restore the stored register values, Task A continue where it left off.
- Control Transfer :- it also take care of the transferring control to the new task (TaskB). Then it means cpu start executes a B instructions.
- Completion :- make everything is ready for the next task Switch.
- Ending the function :- finally it ends its job, and OS continues running on chosen task.

vi. Saving the floating point registers

17 Friday JANUARY 2020

Week 3

Appointments

- Starting with the Intel 80486 processors, a part of the cpu is dedicated to performing complex math operations called Floating-point unit (FPU)
- Intel introduces a mmx instructions to speed up multimedia applications.
- The advantage of this choice is the operating system does not need to handle mmx instructions differently.
- Mmx instructions speed up multimedia applications, because they introduce a single-instruction multiple data for 128-bit registers called the xmm registers.
- The 80x86 microprocessors do not automatically save the FPU, mmx, and xmm registers in TSS.
- To manage this selective saving & restoring of floating-point, mmx and xmm registers there are dataset-features. They stored part of the TSS called tss.i387.

```
if (prev->flags & PF_USEDFPU){
    asm ("fnsave %0": "=m" (prev->tss.i387));
    asm ("fwait")
    prev->flags &= ~PF_USEDFPU;
    stts();
}
```

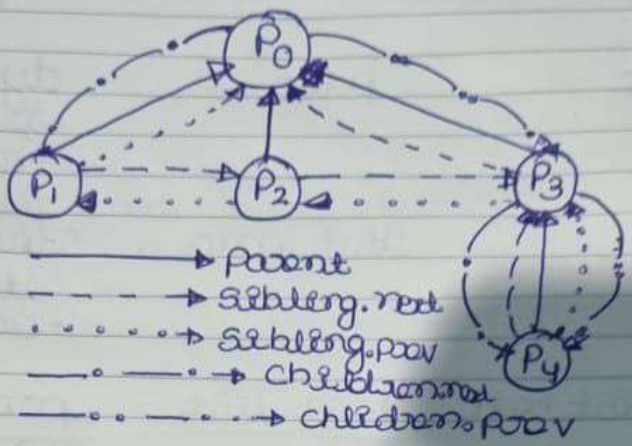
- The stts() macro sets the TS flag of cr0

Go to the core, theories are useless

Appointments

i. Relationships among Processes

- parenthood relationships




i. real_parent :- points to the process descriptor of a process that created P.

ii. parent :- points to the current parent of P. its values coincides with real_parent.

iii. children :- head of the list containing all children created by P

iv. Sibling :- The pointers to the next and previous elements of the Sibling process that have same parent as P.

- Non-parenthood relationships

i. group_leader : process descriptor pointer of group leader p

Cherish your visions and your dreams as they are the children of your soul, the blueprints of your ultimate achievements. - Napoleon Hill

Appointments

Declare_wait_queue_head(name)
int_waitqueue_head()
int_waitqueue_entry(q,p){
q->flags=0;
q->task=p;
q->func=default_wake_function;
}
Define_wait()
add_wait_queue()
add_wait_queue_exclusive()
remove_wait_queue()
wait queue_active()

iii. Process Resource limits

- The resource limits are stored in the current -> signal -> rlim field

```
Struct rlimit {
  unsigned long rlim_cur;
  unsigned long rlim_max;
};
```

Rilimit_AS -> maximum size of address space
" _cpu -> maximum cpu time of process
" _Data -> maximum heapsize in bytes
" _core -> maximum core dump file size
" _Fsize -> maximum file size allowed
" -locks -> maximum number of file locks
" -Stack -> maximum stack size in bytes
" _Rss -> maximum number of page frames owned by process

"The important thing is not being afraid to take a chance. Remember, the greatest failure is to not try." - Debbi Fields

## II Process scheduling

### 1. Scheduling policy :-

- Set of rules used to determine when and how to run new process is called a schedu-ling policy.
- Linux is based on time-sharing techni-que and several process to run on "time multi-plexing" because cpu time is divided in-to two slices. one of the runnable process of course of single process is on one process at given instant of time.
- Scheduling policy is based on ranking proc-ess according to their priority.
- Linux process priority is dynamic. Schedul-ing policy keep track of what doing adjust to their according priority periodically.

### A. Class of process

- There are 3 types of class of process
  i. Interactive process
  ii. Batch process
  iii. Real-time process

1 Wednesday JANUARY 2020

| Appointments | Tasks | Description |
|---|---|---|
| i. | nice() | change of the static priority to conventional process. |
| ii. | setpriority() | set the static priority of group of conventional process. |
| iii. | getpriority() | get the static priority of maximum of group of conventional process. |
| iv. | sched_setscheduler() | set the scheduler policy. |
| v. | sched_getscheduler() | get the scheduler policy of real-time priority of a process. |
| vi. | sched_setparam() | set the real-time priority of a process. |
| vii. | sched_getparam() | get the real-time priority of a process. |
| viii. | sched_yield() | Relinquishes of a conferred process. |
| ix. | sched_get_priority_min() | get the minimum real-time value of a process. |

Your future is hidden in what you do daily - Mike Murdock

Appointments

x. Sched_get_priority max() get the maximum real-time value of a process.

xi. Sched_rr_get_priority() get the real-time round-robin policy.

xii. Sched_setaffinity() set the cpu affinity of mask of process.

xiii. Sched_getaffinity() get the cpu affinity of mask of process.

A. Process preemption

- Linux process is preemptable.
- A process is preempted when its quantum time expires.
- preemption is a fundamental multi-tasking system that ensures fairness, responsiveness and efficient resource utilization.

A. How long quantum last?

- quantum duration is a critical for scation.
- performance : it should be neither too short or long.

Thinking is the hardest work there is, which is probably the reason why so few engage in it. - Henry ford

• The average quantum duration is too short.

2. The scheduling algorithm

• scheduling algorithm used in the earli -er version of linux was quite simple and straightforward. each process is switch to kernel scheduled list of com -patible priorities and select best process to run.
• Scheduling algorithm uses Linux 2.6 was sphisticated.
• Every scheduling process according to their following of scheduling classe -s:-

i. Sched_FIFO :- first - in first out of real-time process when scheduled is assigns to cpu to process and it leaves process descriptor and running it's time of cur -rent position of the runqueue list.
• At the highest priority of the real-ti -me process. and the process uses cpu ti -me as long as wishes. even if other real-time process is having same prio -rity.

ii. Sched_RR :- Round-robin policy process when scheduled assigns cpu to the process it puts in process descriptor at the end of the process in the run

7 Tuesday JANUARY 2020

Week 2

Appointments -runqueues.

- runqueues list has been stored in the run-queues cpu variable.

## Types of runqueues datastructures

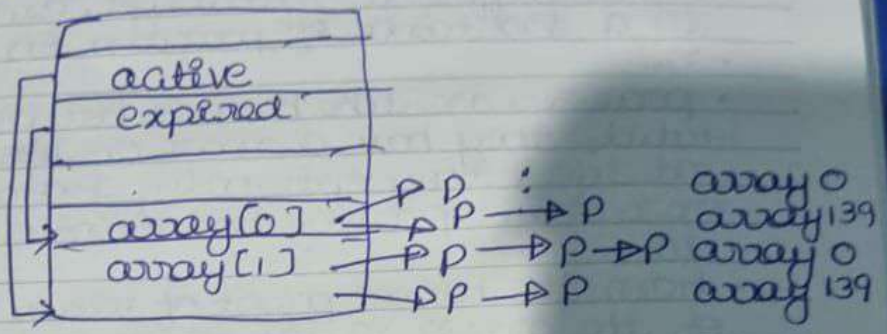
| Types | name | Description |
|---|---|---|
| i. Spin lock | lock_t | protect the list of process. |
| ii. Unsigned long | nr_running | number of runnable process in the list |
| iii. Unsigned long | cpu_load | cpu load factor of a runnable process. |
| iv. Unsigned long | nr_switches | number of switches for the cpu |
| v. Unsigned long | expected_uninterruptable | number of switches for the cpu is to sleep in the uninterruptable |
| vi. Unsigned long | executable_time stamp | Inter relation-ships of the runnable process |

"Always deliver more than expected." - Larry Page

8 Wednesday JANUARY 2020

vii. unsigned long timestamp_last_tick — timestamp of the value time.

- most important of the runqueue data structure is a runnable process.




- The runqueue list is having a two arrays of a each runnable process is having a its dynamic priority.

- The active field of prio_stat array data structure is corresponds to each runnable process in the active field.

- The expired field of prio_stat of array data structure is corresponds to each runnable process in the expired field.

Appointments

- The Intel 80x86 architecture includes a specific segment type called task st -ate segment to store a hardware contex -s.
- There are two main reasons:
  i. when 80x86 cpu switches from user mode to kernel mode it fetches the address of the kernel mode stack from the TSS.
  ii. when a user mode process to attempt to each I/O port by means of in or out instr -uctions.
- Each TSS has its own 8-byte Task State segment descriptor (TSSD). This descri -ptor includes a 32-bit base field poin -ts.
- Type field is set to 11 if the TSSD refers to the TSS of the process currently running on the cpu is set to 9.

iv. Switch_to macro

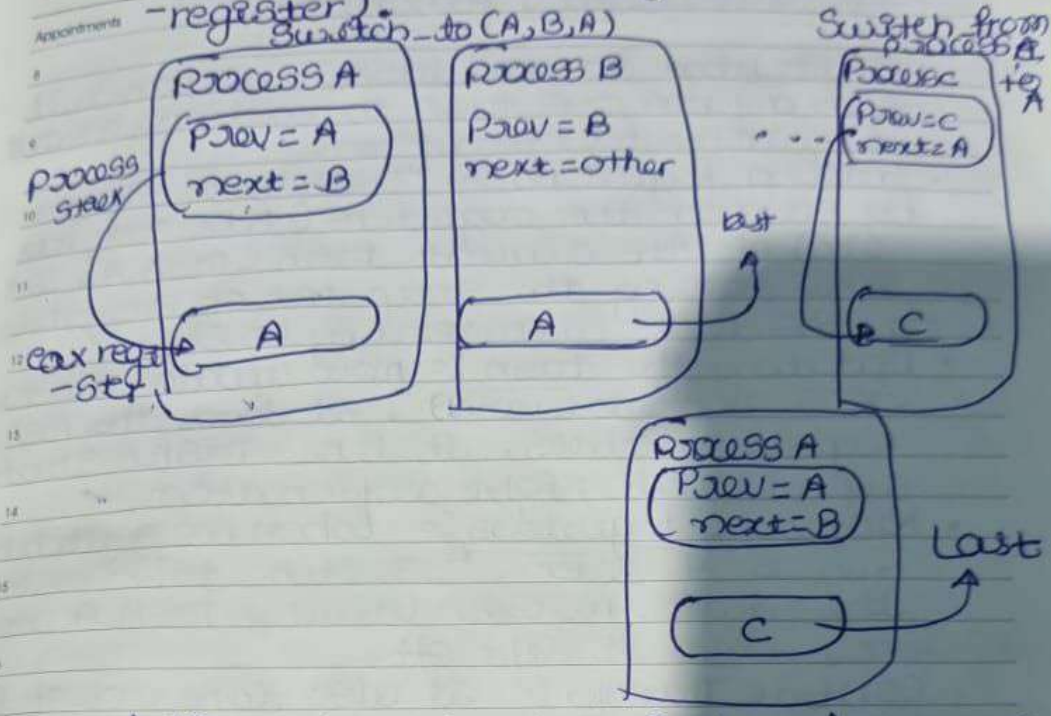
- The switch_to macro typically does on an 80x86 microprocessor by using standard assembly language.
- Save the values of the prev and next in the eax and edx registers respec -tively;

```
movl prev, %eax
movl next, %edx
```

- Saving Flags and EBP: pushfl (to save flags) and pushl %ebp (to save EBP

15 Wednesday JANUARY 2020

- The switch-macro performs a process switch. The switch-macro consist of three parameters, prev, next & last
- macro is invoked by the schedule() -register).




- Updating the stack pointer to previous process : movl %esp, 484(%eax).
- Loading stack pointer to the next process : movl 480(%edx), %esp
- pushing the next process instruction pointer : pushl 480(%edx). This prepares the process to start executing code of the next process.
- Jumping to "_switch_to" function: Jmp_switch_to it essentially hands to control over the next process.
- Return the original process : popl %ebp (to restore EBP) and popfl (to restore flags)

"Opportunities don't happen. You create them." - Chris Grosser

- movl %eax, last

18 Saturday JANUARY 2020

Week - 3 (018-348)

Appointments

i. movl %cr0, %eax ; Load the value of CR0 to EAX

ii. orb $8, %al ; sets the bit (bit3) in AL

iii. movl %eax, %cr0 ; Store the modified value back to CR0

*. Void math_state_restore(void)
{
asm("clts")
if (current → used_math)
asm("frstor %0" : : "m" (current → tss.i387));
else {
asm("fninit");
current → used_math = 1;
}
current → flags |= PF_USEDFPU;
}

19 Sunday

Week - 3 (019-347)

3. Process Descriptor

- Every process Descriptor includes several field related to scheduling.

| Types | Name | Description |
|---|---|---|
| • unsigned long | thread-info → flags | Stores the TIF_NEED_Rescheduled flag |

Great minds discuss ideas, mediocre minds discuss events, small minds discuss people

20 Monday JANUARY 2020

| | | |
|---|---|---|
| • unsigned int | thread_info → cpu | logical number of cpu owning the runqueue |
| • int | prio | dynamic priority of a process. |
| • int | stat_prio | static priority of a process |
| • struct list_head | run_list | pointers to the next previous elements in the runqueue list. |
| • unsigned long | sleep_avg | average sleep time of the process. |

• when a new process is created, sched_fork() is invoked by copy_process(), set the timeslice field of both current process in the following way:-

```
p -> time_slice = (current -> time_slice + 1) >> 1;
current -> time_slice >>= 1
```

You can tell whether a person is clever by his answers. You can tell whether a man is wise by his questions - Naguib Mahfouz

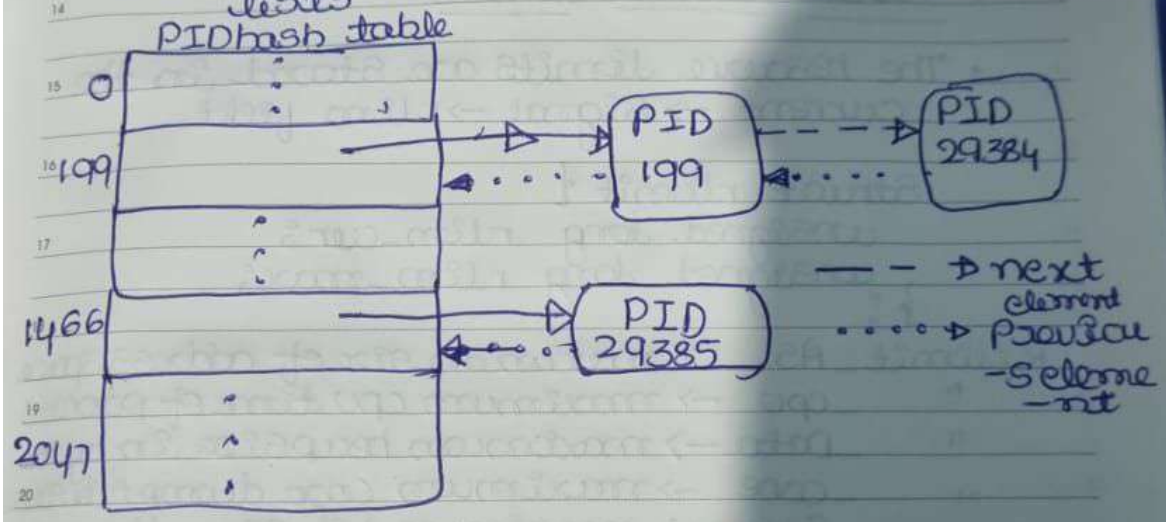
22 Wednesday JANUARY 2020

Appointments

- signal->pgrp :- PID of the group leader for P.
- process descriptor is having different types of PID and each type PID requires its own hash table.

| Hash table type | Field name | Description |
|---|---|---|
| • PIDTYPE_PID | - Pid | - pid of the process |
| • PIDTYPE_PGID | - Pgrp | - Pid of the group leader process |
| • PIDTYPE_TGID | - tgid | - Pid of the thread group leader process |
| • PIDTYPE_SID | - session | - pid of the session leader process. |

- A simple PID hash table and chained lists




## ii. How processes are organized

- Handling wait queues

Service to others is the rent you pay for your room here on earth. - Mohammed Ali

24 Friday JANUARY 2020

## 4. Destroying Process

- process terminates the execution of the code they were supposed to run.
- process terminate is to invoke the exit() function.

### i. Process Termination

- In Linux 2.6 there are two system calls that terminates the user mode application:

i. exit_group() system call :- The main kernel function that implements the system call is called do_group_exit()

ii. exit() system call :- The main kernel function that implements this system call is called do_exit()

iii. The do_group_exit() function :- Kills all the processes belonging to the thread group of current after receiving parameter.

- Steps :-
  - i. checking SIGNAL_GROUP_EXIT flag. If not zero, do_exit()
  - ii. Else setting the SIGNAL_GROUP_EXIT flag.
  - iii. Invoking zap_other_threads() function
  - iv. Invoking do_exit() function

25 Saturday

Week 4 • (025-340)

Christian Lunar New Year (China, Indonesia, Malaysia, Hong Kong and Singapore)

Appointments

• The do_exit() function :- Removes most references to the terminating pro -cess from the kernel data structures. Rece -ive process termination code as param -eter.

• Steps :-

i. Setting PF_EXITING flag.

ii. calling del_timer_sync() function

iii. Detaching data structures

iv. Decreasing usage counters

v. Setting exit_code field

• Invoking exit_notify() function

i. Updating parenthood relationships

ii. checking exit_signal field and if process is last in thread gro -up.

26 Sunday

Week 4 • (026-340)

Republic Day (India), National Day (Australia)

iii. checking exit_signal field and if group has other processes.

iv. Setting exit_state field to EXIT_DEAD and invoking release_task() if process is not being traced.

v. Setting exit_state field to EXIT_ZO -MBIE if exit_signal not equal to 1.

V. Setting PF_DEAD flag.

• Invoking schedule() function.

ii. Process Removal

Steps in release_task() function

"It is not the strongest of the species that survive, nor the most intelligent, but the one most responsive to change." - Charles Darwin

- Decrease a number of processes belonging to user owner of terminated process.
- Removing process from ptrace_children list and assigning parent if being traced.
- Invoking __exit_signal() and exit_notifier()
- Invoking __exit_sighand()
- Invoking __unhash_process()
  i. nr_threads --
  ii. Invoking detach_pid()
  iii. Invoking detach_pid() again if process is thread group leader
  iv. Using Remove_LINK macro.
- Sending signal to notify process death to parent.
- Invoking sched_exit() function
- Invoking put_task_struct(): if counter=0, dropping remaining reference
  - Decrease usage counter and releasing data structure if usage counter=0
  - Releasing process descriptor and memory area.

## 5. Creating Process

- Unix operating systems do, process creation to satisfy user requests. for Example, the shell creates a new process that executes another copy of the shell whenever the user enters a command

i. Modern unix Kernel solve this pro
-blem by introducing three diffe
-nt Kernel mechanisms

• The copy on write technique allows both the parent and the child to read the same physical pages

• Lightweight process: allows both the parent and the child to share many process
-s kernel data structures, such as the paging tables, the open file tables and the signal dispositions.

• Vfork() system call creates process that shares memory address space to paren
-t.

• To prevents the parent from the over
-ting data needed by the child, the parent execution is blocked until chi
-ld exists

ii. The clone(), fork() and vfork() system calls

• fn : specific function to be execut
-ed by the new process

• arg : points to data passed to th
-e fn() function.

• child_stack : specified the usermo
-de stack pointer to be assigned

to the esp register of the child process.

- tls :- meaningful only if the clone-SETTLS flag is set.
- ctid :- meaningful only if the clone-child-settid flag is set.
- clone() :- is actually wrapper function defined in C library.

### iii. The do_fork() function

- The do_fork() function, which handles the clone(), fork(), and vfork() system calls, acts on the following parameters:
  - i. clone_flags :- Same as the flags parameter of the clone
  - ii. stack_start :- Same as the child_stack parameter of the clone.
  - iii. regs :- pointers value of the general purpose registers saved into the kernel mode stack when switching from the usermode to kernel mode.
  - iv. stack_size :- Unused (always set to 0).

Appointments

- parent - tidptr, child - tidptr same as the corresponding ptid and ctid parameters of the clone().