CO2 part A & B 190031187

7. process Table

kernel has a process table that keeps track of all active processes. Each entry in the process table contains pointer to text, data, stack and U Area of process

state field: user running, kernel running etc.

- fields that allow the kernel to locate the process and uArea. Requires while context switch.
- process ID
- user ID
- process size
- Scheduling parameters
- signal field
- various timers

UArea is the extension of process table entry

- pointer to process table entry
- The current directory and current root
- The user file descriptor table
- Limit fields

  Restrict process size

  Restrict size of the file it can write

- Control terminal field

[illegible]

190031187

9.
```
#include <stdio.h>
#include <stdlib.h>

#include <sys/types.h>
#include < sys/wait.h>

#define LONG_SLEEP 20
#define SHORT_SLEEP 10
#define N_CHILD 3

int main (int argc, char *argv[])
{
      int i=0;
      Pid_t pid
      int exit_status;

      printf("parent: starts\n");
        for (i=1; i<N_CHILD; i++){
           if (!(pid = fork())){
               printf("chid: starting %d\n", i);
               fflush(NULL);
               sleep(SHORT_SLEEP);
               printf("child: ending %d\n",i);
               fflush(NULL);
               exit(i);
           }
           printf("parent: forked child %d with pid %d\n",
                                        i, pid);

           fflush(NULL);

      }
      sleep(LONG_SLEEP);
      printf("parent-reaps\n");
for ( i=1; i<N_CHILD ; i++ )
{
     pid = wait (&exit_status) ;
     printf ("parent : child %d has exit code %d\n",
                    pid, WEXITSTATUS (exit_status));

}
sleep(SHORT_SLEEP);
printf(" parent : exists \n");
exit(0);

}
```

10. (i) Flow of Context Switch

1. invoke the interrupt (ex :- Timer Interrupt (trap.c 109))
2. invoke the Scheduler (Yield (proc.c 390))
3. Start the Context switch (swtch (proc.c 387))

Before context switch interrupt happens

trapasm.S

→ Segment register saves physical address of the segment offset.

```
.globl alltraps

alltraps :
        pushl %ds #data segment

register
        pushl %es

register
        pushl %fs

register
        pushl %gs

register
        pushal # all local registers
```

→ Set up data segments

→ Then in trap.c , calling yield() to invoke scheduler.

Also, after invoking scheduler, the system starts context switch said before in

swtch (struct context **old, struct context *new)

190031187

→ context switch starts

Context switch writes in assembly and links by linker

swtch.S

```
swtch :
    movl  4(%esp) %eax  # address of
** old  !! little endian
    movl  8(%esp)

  # save all callee registers
    pushl  %ebp
    pushl  %ebx
    pushl  %esi
    pushl  %edi

    movl  %esp, (%eax)
  means *old

    movl  %edx, %esp
   load
  # new callee-save
  registers
  this is the return address

    popl  %edi
    popl  %esi
    popl  %ebx
    popl  %ebp
    ret
```

190031187

10/11j

process context

The context of a process is its state:

- Text, data, register
- process region table, UArea
- user stack and kernal stack

- when executing a process, the system is said to be executing in the context of process

Fork system call

→ It helps user to create a new process

→ The process which invoked fork is called parent process

→ The newly created process is called child process

→ syntax of fork system call is

pid = fork()

→ In parent process pid will have its child PID and int child pid will have 0.

Tasks perfolmed

1. It allocates a slot in the process table
2. It assigns unique ID number to child process
3. It makes a logical copy of the context of the parent process
4. It increments file and mode table counters for file associated with the process.
5. It returns the ID number of child to the parent process, and 0 value to child process.