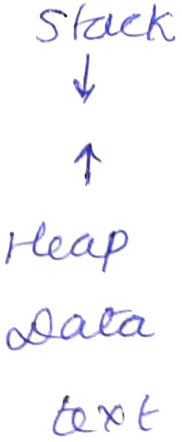
23/11/2020

Assignment-2

① what is process management. what are the importance?.

A process is a program in execution. when we actually run the binary code, it becomes a process.

A process is an active entity, as opposed to a program, which is considered to be a passive entity. A single program can create many processes when run multiple times.



A process management involves various tasks like creation, scheduling, termination of processes and a dead lock. process is a program that is under execution, which is an important part of Os.

The OS should allocate resources that enable processes to share and exchange information. It also protects the resources of each process from other method and allows synchronization among process.

Stack : Stack stores temporary data like fn parameters returns addresses and local variable.

Heap : allocate my, which may be processed during its run time.

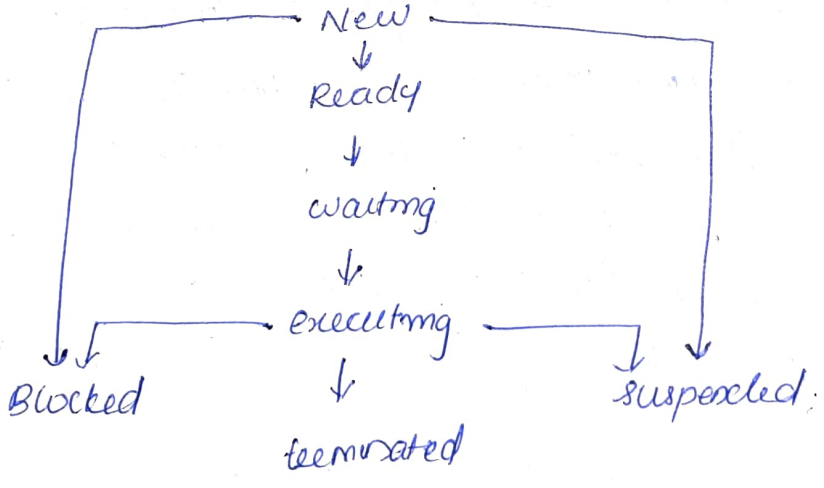
Data : contains variable.

Text : includes current activity, which is represented by the value of program counter.

Process control block (PCB) is the data structure maintained by OS for every process. The PCB should be identified by an integer process ID. It helps you to store all info required to keep track of all running process.

It is also accountable to store the content of processor registers. This saves when process moves from running & returns back to it. The info will quickly acid to PCB when process makes state transition.

4. With neat diagram, explain different states in process?



A process state is a condition of the process at a specified instant of time. It also defines the current position of process.

There are mainly 7 stages:

(1) New: The new process created when a specific pgm calls from secondary / hard disk to 1° mly / RAM.

(2) Ready: In ready, the process should loaded to 1° mly which is ready for execution

(3) Waiting: The process waits for allocation of CPU time & other resources

(4) Executing: The process is executing

(5) Suspended: The process is ready for execution but has not placed in ready queue by OS.

(6) Terminated: The time when process is terminated. After completing all process, & Resources used by process, the mly becomes free.

5. What is context switching? Explain the scenarios where context switching happens?

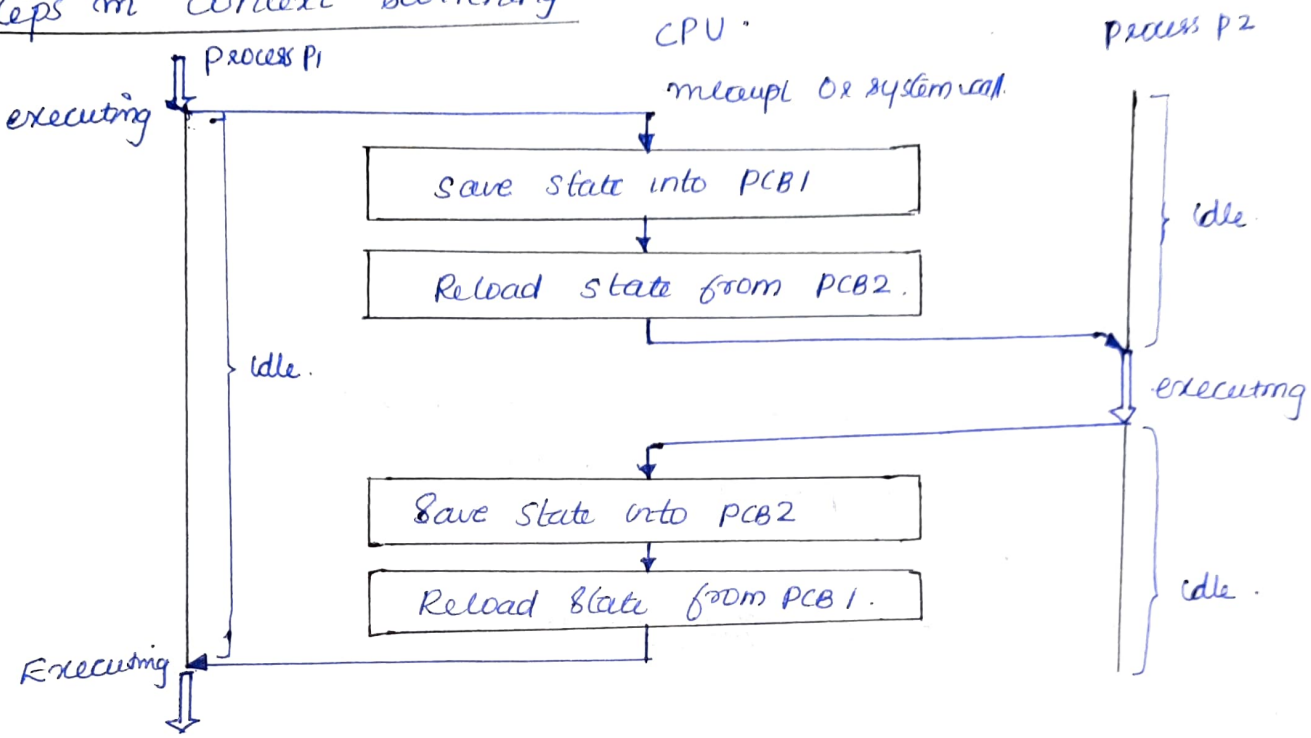
Context switching is procedure that a computer's CPU follows to change from one task (or process) to another while ensuring that the tasks do not conflict. Effective context switching is critical if a computer is to provide user friendly multitasking.

But context switching involves a number of steps. The process should save the context of that process. If not saving the context of any process P then for some time, when the process P comes in the CPU for execution again, then the process will start executing from the starting.. so the process should save.

context ~~should~~ switching can happen due to following reasons:

- When process of high priority comes in ready state. Then the exis stopped and high priority process should be given CPU for execution
- When interrupt occurs the the process in running state should stoped and the interrupt should happen
- When transition between kernal mode and user mode required

Steps in context switching



① context of process P_1, running state saved in PCB P_1

② move PCB1 to relevent queue, ie ready queue, I/O queue, waiting queue etc.

③ From ready state, select the new process to be process P_2

④ PCB2 by setting the process state to running. If P_2 executed earlier, It will resume execution of P_2.

⑤ Repeat the same step.

The time envolved for context switching is called context switching time.

Advantage

Context switching is to achieve multitasking.

context switching is so fast that user feels the CPU is executing more than 1 task at a time.

(5) What is an orphan child process? Give example code to explain?

A process whose parent process no more exist, i.e either finished or terminated w/o waiting for its child process to terminate is called an orphan process.

```
#include <stdio.h>
int main()
{
  int pid;
  pid = getpid();
printf("current process id is %d\n", pid);
printf("forking child process...\n");
  pid = fork();
printf("pid after fork %d", pid);
if (pid<0)
  { exit(-1);
  }
else if (pid==0)
printf("\n ~~orphan~~ Child is sleeping..");
  sleep(5);
}
else
    { printf("parent process completed..");
    }
  return 0;
}
```

(7). What is a zombie process? Give an example code to explain?

A process which has finished the execution but still has entry in PCB to report to its parent process is known as zombie process. A child process always first become a zombie before being removed from process table.

```
#include <stdio.h>
#include <sys/types.h>
#include<unistd.h>
#include <stdlib.h>

int main().
{
   int pid;
   pid = getpid()
   printf("current process id : %d \n", pid);
   pid = fork();
   if (pid<0)
     { exit (-1);
     }
   else if (pid ==0)
    { printf(" child process started \n");
      printf (" child process completed \n"); }
   else
    {
      sleep (10);
     printf ("parent process running \n");
     printf (" I am in zombie state \n");
     while (1)
       {
       /* infinite loop */
       }
   }
  return 0;
}
```

(8) What is the difference blw Orphan child process and zombie process Explain with example.

Zombie	Orphan
→ execution complete, but it still has entry in process table	→ still running if child process terminates.
→ didn't use any sim resource but retains their process ID.	→ orphaned process run in background without any manual support.

```
#include <sys/types.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main()
{
  int pid;
pid = fork();
if (pid == 0)
  {
   printf("The child process complete\n");
  }
else
(pid>0)
else
  {
   sleep(10);
printf("I am in zombie");
  while(1)
  {
  }
}
return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main()
{
  int pid;
  pid = fork()
if(pid == 0)
  {
  sleep(10);
  }
else
  {
  printf("in parent\n");
  }
  return 0;
  }
```

3698688

(9) What is an INODE ? Explain with neat diagram.

In unix based OS, each file is indexed by an Inode. Inode are special disk blocks they are created when file s/m created.

NO. of inode limits the total number of files / directories.

→ contains.

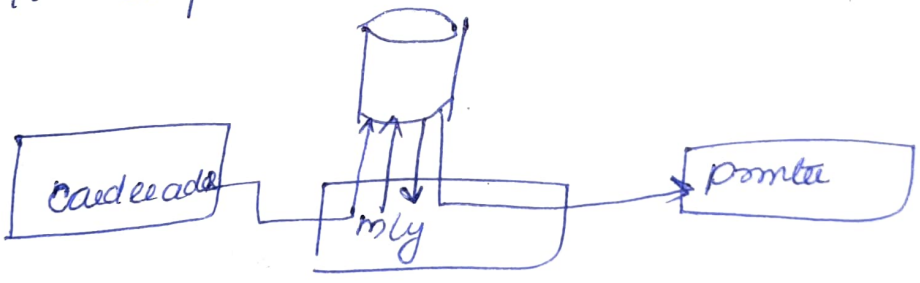
- Administrative information
- number of direct block.
- Single indirect pointer → used as index block. If file is too big

Spooling is an acronym for simultaneous phenpheral Operations on line. Spooling refers to putting data of various I/O jobs in a buffer. This buffer is a special area in memory or hard disk which access to I/O devices

→ Handles I/O device data spooling as device having different data access rates.

→ maintains the spooling buffer which provides a waiting station where data can rest while the slower device catches up.

→ maintains llel computation; because I/O in llel fashion.



Advantage:

. Spooling operation uses a disk as a very large buffer.

. Spooling is capable of overlapping I/O operations for 1 job with process operation for other job

Swap space

→ Swap my on disc, when we need more my.
→ Space of harddisc, substitute of physical my.
→ considered as virtual my, which has process my images.
→ swap space help computer OS in pretending that it have more ram than it actually has.
→ also called swap file
→ Interchange of data between virtual my and real my are called swaping & space on disc is swap space.
→ Swap space is portion of virtual my on harddisc, when used RAM is full.

Uses

→ can be used as single contigenous my which reduces I/O operations to read or write a file
→ applications which are not used or are less used can be kept in swap file
→ having sufficient swap file, help some physical my free all time.

Depends on user, whether he wants to use virtual my or not.

26-12-2020.

① What is a NULL Pointer?

A Null pointer that doesnot point to any memory location. It stores the lease address of the segment. points / store the null value where void is type of pointer.

Specially defined in std def header file. means refereeing to 0th memory location.

→ used to initialise 0 pointer variable when pointer does not point to valid mly address
→ used to perform error handling with pointer before dereferencing pointer
→ passes as fn argument when do not want to pass actual mly addess.

```
# include <stdio.h>
int main()
  { int *ptr;
    ptr = (int *) malloc (4 * sizeof (int));
  if(ptr == NULL)
    { printf ("mly is not allocated");
    }
  else
    { printf ("mly allocated");
    }
  return 0;
  }
```

② What is the size of a pointer variable?

It is not fixed. It depends on different issues like OS, CPU architecture etc..

In 32 bit machine, size is 4 bytes
On 64 bit " , size is 8 byte.

Regardless of what type of they are pointing, they have fixed size.

③ What is Dangling pointer ?

pointer pointing to a memory location, that has been deleted (or freed) is called dangling pointer. 3 different ways:

① . De allocation of memory.

```
int main()
  {
   int *ptr = (int *) malloc (sizeof(int));
   free (ptr); // now ptr becomes dangling pointer.
   ptr = NULL; (no more dangling).
  }
```

② . Function call.

```
int *fn().
 { int x=5          // local variable is not static.
   return &x;
 }.
int main ()
 { int *p = fn().
   fflush (stdin); // pointer points to something which is not valid anymore.
   Printf ("%d", *p);
   return 0;
 }
```

O/p: A garbage address.

3. Variable out of scope.

```
Void main ()
{ int * ptr;
  . . . .
  { int ch;
    ptr = &ch.
  }
  . . . .
```

// Here ptr is dangling ptr.

(4) What is a file pointer? What are the functions involved with file pointer?

File pointer is used to handle and keep track on the files being accessed. A new data type called FILE is used to declare file pointer.

Functions that helps to perform basic file operations:

fopen () - Create a new file or open a existing file.
fclose () - Closes a file.
getc () - reads a character from a file.
Putc() - writes a character to a file
fscanf() - reads a set of data from a file.
fprintf() - writes a set of data to a file
getw() - reads a integer from a file.
putw() - Writes a integer ~~from~~ to a file.
fseek() - Set the position to desire point.
ftell () - gives current position in the file.
rewind() - set position to the begining point

What is the return of malloc fns. what type casting is used with malloc?

void * malloc (size_t size) - allocates the requested mry and returns a pointer to it.

Return values :- The fns returns a pointer to the allocated mry, or NULL if the requested fails.