ot. No.

Page No.
Date. 10/7/19

Day 1 :- Self Assessment with General Purpose Utilities in UNIX-like Systems

Important commands

- mkdir → create directory
- cd → change directory
- pwd → present working directory
- ls → shows content of present directory.
- ls -al → lists all the hidden file.
- echo → used to print in terminal
  echo $$ (prints terminal ID)
  ~~echo $HOME~~
  echo $HOME (prints the path to home directory)

A code to demonstrate System calls and child processes

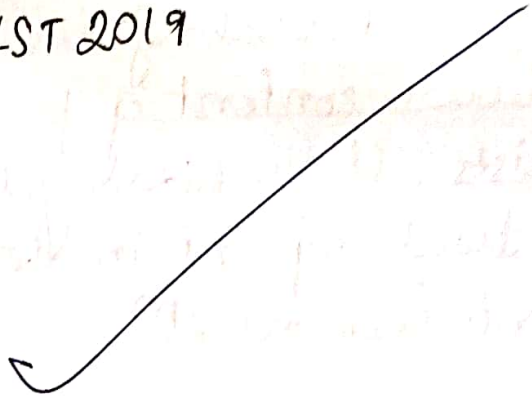
```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <unistd.h>
```

P.T.O

Faculty Signature .....

Output

```
$ ./a
Parent: my pid is = 4975
Parent: my parent pid = 3688
Parent: my childPID = 4976
Parent: N do waiting
child : my pid is = 4976
child : my parent pid is = 4975
child : I am sleeping for sometime
Wed Jul 10 10:49:15 IST 2019
child: good bye
Parent: goodbye.
```



Expt. No.

Page No.
Date.

```
Void main()
{
    pid_t childpid;
    childpid = fork();
    if (childpid >= 0)
    {
        if (childpid == 0)
        {
            printf("child: my pid is = %d\n", getpid());
            printf("child: my parent pid is = %d\n", getppid());
            printf("child: I am sleeping for sometime\n");
            sleep(5);
            system("date");
            printf("child: good bye\n");
        }
        else
        {
            printf("Parent: my pid = %d\n", getpid());
            printf("Parent: my parent pid is = %d\n", getppid());
            printf("Parent: my child id = %d\n", childpid);
            printf("Parent: Now waiting\n"); wait(NULL);
            printf("Parent: goodbye\n"); }}
        else
            printf("Error in child creation\n"); }
```

Teacher's Signature ..... Date. 12/7/19

Expt. No.

Page No.
Date.

Day 2 →

Important commands:

grep ⇒ Print lines matching a pattern
cut : Remove sections from each line of files.
sort : Sort lines of text files.
sed : Stream editor for filtering and transforming text
tr : Translate or delete characters
cat : Concatenate files and print on the standard output
cp : Copy files and directories.
cmp : Compare two files byte by byte
comm : Compare two sorted files line by line
mv : move file
rmdir : Remove empty directories
ps : Report a snapshot of the current processes
chmod : Change mode
w : Shows the users currently on the machine and the processes.
whoami : Shows username of the current user.
who am i : shows the current username
gzip : Compresses files. Each single file is compressed into a single file
gunzip : Compress or expand a file or list of files.

Teacher's Signature .....

Output:

```
$ chmod 764 fun
$ ./fun
$ your good name please Saikat
$ your age please 20
$ Hello Saikat, next year you will be 21 years old.
```

Expt. No.

Page No.
Date.

ls -ahl ./usr/bin | tr -s ' ' | cut -d ' ' -f9
⇒ shows the 9th column of the list.

ps -Hef | head -15 | tail -5 ⇒ shows the last 5 lines of the first 15 line of the process list

$ Sayhello() ⇒

```
$ Sayhello()
{
  echo $LOGNAME, how are you doing?
  return
}
$ Sayhello. ↵
$ Jyaga2, how are you?
```

```
$ gedit fun          // fun file
echo .your good name please
read name
echo .your age please
read age
newage = `expr $age +1`
echo Hello.$name, next year you will be $newage
-old
```

24/7/17
Teacher's Signature .....

OXFORD

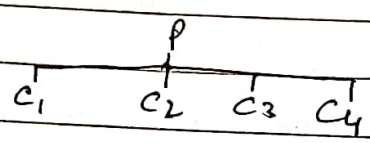
Expt. No.

Page No.
Date.

Day 3

1) Write a program that would create 4 child process from the parent process to implement Degree of multiprogramming in a uniprocessor system. The child process will create a directory structure in your bash shell.

Case 1:



Code:-

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
void main() {
  pid_t c1, c2, c3, c4;
  system("rm -r p");
  system("mkdir p");

  if ((c1 = fork()) == 0) {
    printf("child (%d): my parent = %d \n", getpid(), getppid());
    system("mkdir p/c1");
    sleep(2);
    printf("child (%d): Good bye \n", getpid());
  }
```

Teacher's Signature .....

Expt. No.

Page No.
Date.

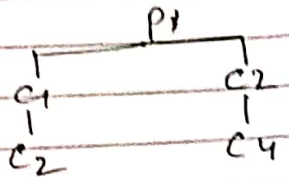
```
else if (c1>0)
{   printf("Parent (%d): Creating four child process at the same
            level", getpid());

    if ((c2=fork())==0) {
    printf ("Child (%d): Parent = %d \n", getpid(), getppid());
    System ("mkdir p/c2");
    sleep(2);
    printf("Child(%d): Good bye.\n", getpid());
    }
    else if (c2>0){
      printf if ((c3=fork())==0){
      printf ("Child (%d): Parent = %d \n", getpid(), getppid());
       System ("mkdir p/c3");
       sleep(2);
       printf ("Child (%d): Good bye \n", getpid());
    }
else if (c3>0) {
    if ((c4=fork())==0) {
     printf ("Child (%d): my parent : %d \n", getpid(), getppid());
     system ("mkdir p/c4");
     sleep(2);
     printf ("Child(%d): Goodbye \n", getpid());
    } else if (c4>0) { sleep(10);
    System ("ps -eaf | grep pts/0 | grep -c dey 3-i");
    wait (NULL);
    printf ("Parent (%d): finally I am exiting \n");
}}}
```

Teacher's Signature .....

Expt. No.

Page No.
Date.

Case 2:




```
#include <stdio.h>
#include <sys/type.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
void main(){
pid_t C1, C2, C3, C4;
 system("rm -r P1");
 system("mkdir P1");
if((C1 = fork())==0){
  system("mkdir P1/C4");
  if((C2 = fork())==0){
  printf("Child (%d): my parent (%d)\n", getpid(), getppid());
  system("mkdir P1\C1\C2");
  sleep(2);
  printf("Child(%d): Goodbye.\n", getpid());
}
else if(C2>0){
  printf("Child (%d): my parent (%d)\n", getpid(), getppid());
  sleep(2);
  printf("Child(%d): Goodbye\n", getpid());
}}
```

Teacher's Signature .....

Expt. No.

Page No.
Date.

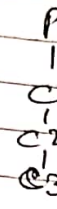
```
else if (c1>0) {
  printf("Parent (%d): Creating four child at 2 level \n", getpid());
  if((c3 = fork()) == 0) {
    printf("Child (%d): my parent (%d)\n", getpid(), getppid());
    System("mkdir P1/C3");

    if((C4 = fork()) == 0) {
      printf("Child (%d): my parent(%d)\n", getpid() getppid());
      system("mkdir P1/C3/C4");
      sleep(2);
      printf("Child (%d): Good bye.\n", getpid());
    }
    else if (c4>0)
    {
      sleep(2); printf("Child (%d): Good bye.\n", getpid());
    }}
  else if (c3>0){
  sleep(10);
  system("ps -ef | grep pts/0 | grep day3-i");
  wait(NULL);
  printf("Parent (%d); finally I am exiting\n", getpid());
}
```

Teacher's Signature .....

Expt. No.

Page No.
Date.

Case: 3




```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
void main().
{
    pid_t C1, C2, C3, C4;
    System("rm -r P2");
    System("mkdir P2");
    if((C1=fork())==0){
    printf("Child(%d): my Parent(%d)\n", getpid(), getppid());
    System("mkdir P1/C1");
    sleep(2);
    printf("Child(%d): Goodbye.\n", getpid());
    }
else if((C2=fork())==0){System("mkdir P2/C1/C2")
    printf("Child(%d): my parent(%d)\n", getpid() getppid());
    sleep(2);
    printf("Child(%d): Goodbye\n", getpid());
    }
}
```

Teacher's Signature .....

Expt. No.

Page No.
Date.

```
if ((c3 = fork()) == 0){
    printf("Child (%d) : my parent = %d\n", getpid(), getppid());
    system("mkdir p2/c1/c2/c3");
    if ((c4 = fork()) == 0){
        printf("Child (%d): my parent : %d\n", getpid(), getppid());
        system("mkdir p2/c1/c2/c3/c4");
    }
    else if (c4 > 0){
        sleep(2);
        printf("Parent (%d): Goodbye\n", getpid());
    }
    else if (c3 > 0)
    { sleep(2);
    printf("Parent (%d): Goodbye\n", getpid());
    }
else if (c2 > 0){
    sleep(2);
    printf("Parent (%d): Goodbye\n", getpid());
    }
else if (c1 > 0){ sleep(2);
system("ps -eaf | grep pts/0 | grep day3.3");
wait(NULL);
printf("Parent (%d): finally I am exiting");
}
}
```

Teacher's Signature .....

Expt. No.

Page No.
Date.

#Shell.sh#

#!/bin/sh
read name
echo "First shell program of $name
Program $0
The number of arguements is $#"
The arguements are $*"
echo "bye $LOGNAME"

Terminal

$ ./Shell.sh day3-1.c day3-2.c day3-3.c

$ Saikat
First shell program of Saikat
Program ./Shell.sh
The number of arguements is 3
The arguments are day3-1.c day3-2.c day3-3.c

bye 2yr cgr2.

Sai
7/8/19.

Teacher's Signature .....

Output :/a

Child: my id is (4201) with a parent (4200)
Parent (4200) is capturing child exit = 4201.

Expt. No.

Page No.
Date.

Day 4:- Shell Programs and a child process program

Program 1

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

void main()
{
    pid_t pid, pid1;

    if((pid = fork()) == 0)
    {
    printf("Child: my id is (%d) with a parent (%d)\n", getpid(),
                                                        getppid());
    }
    else if (pid > 0)
    {
    pid1 = wait(NULL);
    if(pid == pid1)
{ printf("Parent (%d) is capturing child exists = %d\n", getpid(), pid1);
    }
}
}
```

Teacher's Signature .....

Output.

```
$ sh shell1.sh myfile

the shell file running shell2.sh
1 hi my
2 name is
3 Saikat Kundu
4 CSE - A
5 Roll - 60

Total 5 lines read from myfile.
```

Output

```
$ sh shell2.sh

first index : abc
fifth index : csen 3113
first method : abc edf xyz mno csen 3113
second method : abc edf xyz mno csen 3113.
```

Expt. No.

Page No.
Date.

Shell1.sh

```
#!/bin/bash

echo the shell file running $0
filename = $1.
while read line
do
    let count++
    echo "$count $line"
done < $filename .
echo Total $count lines read from $filename.
```

Shell2.sh

```
#!/bin/bash
name[0] = "abc"
name[1] = "edf"
name[2] = "xyz"
name[3] = "mno"
name[4] = "csen 3113"

echo first index : ${name[0]}
echo fifth index : ${name[4]}
echo first method : ${name[*]}
echo second method : ${name[@]}.
```

Teacher's Signature .....

Output

```
$ sh shell3.sh
0
1
2
3
4
5
```

Output

```
$ sh shell4.sh

Welcome
Your username : 2yrgr2
Identify username : 2yrgr2
Current date and time : Wed Aug 7 16:34:56 IST 2019
Today is identified by 6 parameters : Wed Aug 7 16:34:56 IST 201
Currently logged in users:
2yrgr2 tty7        2019-08-07 14:36 (:0)
2yrgr2 pts/3       2019-08-07 16:06 (:0.0)

user login data:
Wed Aug 7 16:06 still logged in.
```

Expt. No.

Page No.
Date.

shell 3.sh

```
#!/bin/bash

a=0
while [ $a -ge 0]
do
    echo $a
    if [ $a -eq 5]
    then
        break
    fi
        a=`expr $a + 1`
done
```

shell 4.sh ⇒

```
#!/bin/bash
echo Welcome
echo Your Username : $USER.
echo Identify username : $LOGNAME
echo Current date and time : $(date)
set `date`
echo .Today is identified by 6 parameters : $1 $2 $3 $4 $5 $6
echo Currently logged in users:
who
echo user login data:
last $LOGNAME|head -1|tr -s ' '|cut -d " " -f4-11
```

Teacher's Signature .....