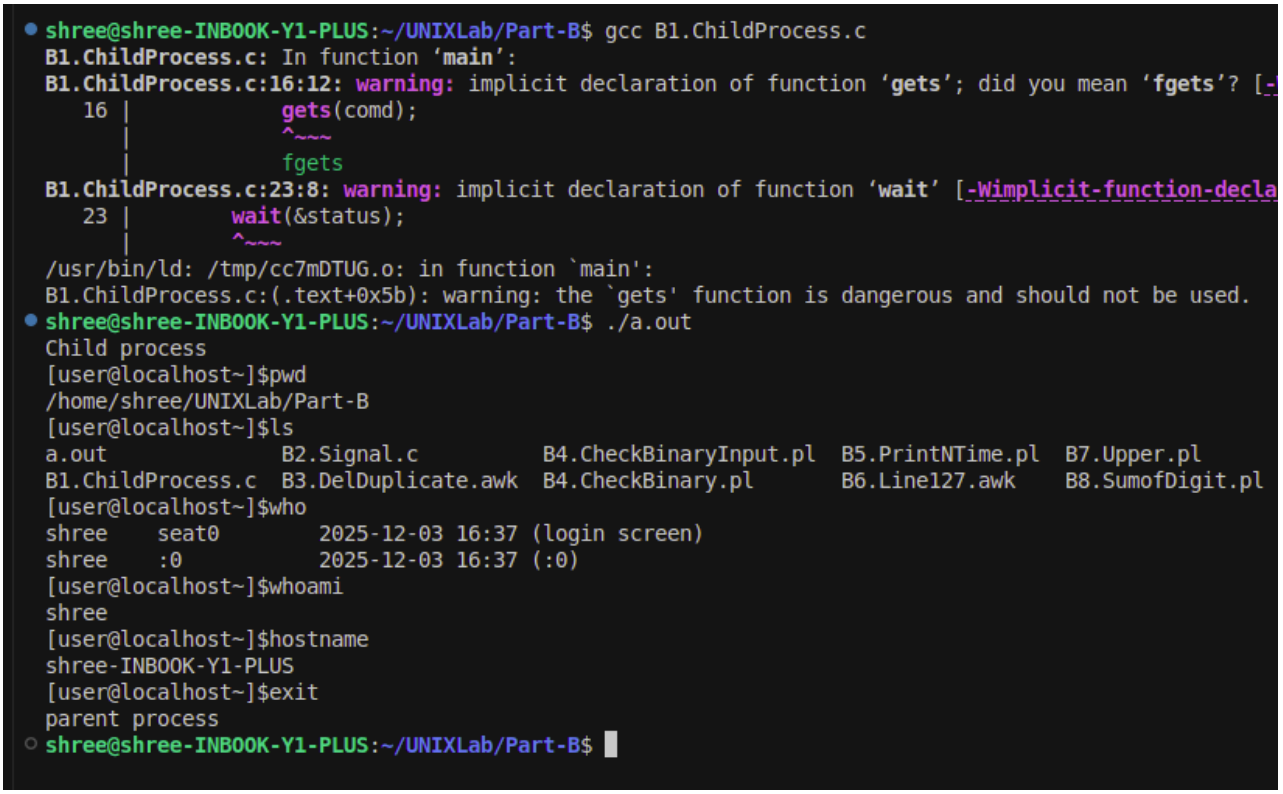
Linux - Unix Shell Programming Lab Programs

B1. Write a C program that creates a child process to read commands from the standard input and execute them (a minimal implementation of a shell – like program). You can assume that no arguments will be passed to the commands to be executed.

B1.ChildProcess.c

```c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>
int main()
{
  int pid,status;
  char comd[20];
  pid=fork();
  if( pid == 0 )
  {
      printf("Child process\n");
      while( strcmp(comd,"exit")!= 0 )
      {
          printf("[user@localhost~]$");
          gets(comd);
          system(comd);
      }
      exit(0);
  }
  else
  {
      wait(&status);
      printf("parent process\n");
  }
}
```


shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ gcc B1.ChildProcess.c
B1.ChildProcess.c: In function 'main':
B1.ChildProcess.c:16:12: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-
16 | gets(comd);
| ^~~~
| fgets
B1.ChildProcess.c:23:8: warning: implicit declaration of function 'wait' [-Wimplicit-function-decla
23 | wait(&status);
| ^~~~
/usr/bin/ld: /tmp/cc7mDTUG.o: in function `main':
B1.ChildProcess.c:(.text+0x5b): warning: the `gets' function is dangerous and should not be used.
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ ./a.out
Child process
[user@localhost~]$pwd
/home/shree/UNIXLab/Part-B
[user@localhost~]$ls
a.out B2.Signal.c B4.CheckBinaryInput.pl B5.PrintNTime.pl B7.Upper.pl
B1.ChildProcess.c B3.DelDuplicate.awk B4.CheckBinary.pl B6.Line127.awk B8.SumofDigit.pl
[user@localhost~]$who
shree seat0 2025-12-03 16:37 (login screen)
shree :0 2025-12-03 16:37 (:0)
[user@localhost~]$whoami
shree
[user@localhost~]$hostname
shree-INBOOK-Y1-PLUS
[user@localhost~]$exit
parent process
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$

B2. Write a C Program to register a signal handler for SIGINT and when it receives the signal, the program should print some information about the origin of the signal.

B2.Signal.c

```c
#include<stdio.h>
#include<signal.h>
void sig_handler(int signo)
{
  printf("Signal caught is %d\t",signo);
}


int main(void)
{
  (void) signal(SIGINT,sig_handler);
  while(1)
  {
      printf("Hello world\n");
      sleep(1);
  }
  return 0;
}
```

```
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ gcc B2.Signal.c
B2.Signal.c: In function ‘main’:
B2.Signal.c:15:8: warning: implicit declaration of function ‘sleep’
   15 |        sleep(1);
      |        ^~~~~
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ ./a.out
Hello world
Hello world
^CSignal caught is 2    Hello world
Hello world
^CSignal caught is 2    Hello world
Hello world
^Z
[6]+  Stopped                 ./a.out
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$
```

B3. Design an Awk script to delete duplicated lines from a text file. The order of the original must remain unchanged.

B3.DelDuplicate.awk

```
BEGIN{
    n=1
}
{
    a[n++]=$0
}
END{
    for(i=1;i<n;i++)
    {
        flag=1
        for(j=1;j<i;j++)
            if(a[i]==a[j])
                flag=0;
        if(flag==1)
            printf("%s \n",a[i])
    }
}
```

```
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ cat > data.txt
Line 1
Line 2
Line 1
Line 3
Line 2
^Z
[7]+  Stopped                 cat > data.txt
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ awk -f B3.DelDuplicate.awk  data.txt
Line 1
Line 2
Line 3
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$
```

B4. Implement a PERL script that takes file as an argument, checks whether file exists and prints binary if file is binary.

B4.CheckBinaryInput.pl

```
#!/usr/bin/perl
printf("Enter the Filename:");
chop($f=<STDIN>);

if(-e $f)
{
   if(-B $f)
   {
       printf("$f is a BINARY FILE \n");
   }
   else
   {
       printf("$f is NOT a Binary File \n");
   }
}
else
{
   printf("$f doesn't Exist\n");
}
```

```
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ ls
a.out            B2.Signal.c          B4.CheckBinaryInput.pl  B5.PrintNTime.pl  B7.Upper.pl       B9.Split15.awk  ic.jpg
B1.ChildProcess.c  B3.DelDuplicate.awk  B4.CheckBinary.pl       B6.Line127.awk    B8.SumofDigit.pl  data.txt        long_line
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ perl B4.CheckBinaryInput.pl
Enter the Filename:ic.jpg
ic.jpg is a BINARY FILE
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ perl B4.CheckBinaryInput.pl
Enter the Filename:data.txt
data.txt is NOT a Binary File
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ perl B4.CheckBinaryInput.pl
Enter the Filename:You.txt
You.txt doesn't Exist
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$
```

Method 2 :

B4.CheckBinary.pl

```
foreach $f (@ARGV)
{
    if(-e $f)
    {
        if(-B $f)
        {
            printf("$f is a BINARY FILE \n");
        }
        else
        {
            printf("$f is NOT a Binary File \n");
        }
    }
    else
    {
        printf("$f doesn't Exist \n");
    }
}
```

```
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ ls
a.out             B2.Signal.c          B4.CheckBinaryInput.pl  B5.PrintNTime.pl  B7.Upper.pl       B9.Split15.awk  ic.jpg
B1.ChildProcess.c  B3.DelDuplicate.awk  B4.CheckBinary.pl       B6.Line127.awk    B8.SumofDigit.pl  data.txt        long_line
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ perl B4.CheckBinary.pl ic.png
ic.png doesn't Exist
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ perl B4.CheckBinary.pl ic.jpg
ic.jpg is a BINARY FILE
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ perl B4.CheckBinary.pl data.txt
data.txt is NOT a Binary File
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$
```

B5. Prompt user to input the string and a number, and prints the string that many times, with each string on separate lines using PERL script.

B5.PrintNTime.pl

```
#!/usr/bin/perl
printf("Enter the String:");
$a=<STDIN>;
printf("Number of times it should be displayed:");
chop($b=<STDIN>);
$c=$a x $b;
printf("Result is: \n$c");
```

```
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ perl B5.PrintNTime.pl
Enter the String:Hello Nitte
Number of times it should be displayed:3
Result is:
Hello Nitte
Hello Nitte
Hello Nitte
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$
```

B6. Design an Awk program to provide extra symbol (i.e. * or @) at the end of the line (if required) so that the line length is maintained as 127.

B6.Line127.awk

```
{
  y = 127 - length($0)
  printf($0);
  if (y > 0 )
      for(i = 0;i < y; i++)
          printf("*");
      printf("\n");
}
```

```
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ cat > LongText.txt
An Awk program to pad the end of a line with '*' symbols (if required) so that the total line length is 127 characters.
^Z
[9]+  Stopped                 cat > LongText.txt
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ awk -f B6.Line127.awk LongText.txt
An Awk program to pad the end of a line with '*' symbols (if required) so that the total line length is 127 characters.********
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$
```

B7. Implement a PERL script that prints its command line argument, one per line after translating all lower-case letters to uppercase.

B7.Upper.pl

```
#!/usr/bin/perl
die("you have not entered the arguments\n")if(@ARGV==0);
foreach $arg(@ARGV)
{
  $arg=~tr /a-z/A-Z/ ;
  printf("$arg\n");
}
```

```
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ perl B7.Upper.pl unix programming lab
UNIX
PROGRAMMING
LAB
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$
```

B8. Find the sum of digits of an unsigned number passed through argument using PERL.

B8.SumofDigit.pl

```
foreach $num (@ARGV)
{
   $original_no=$num;
   $sum=0;
   until($num==0)
   {
       $digit=$num%10;
       $sum=$sum+$digit;
       $num=int($num/10);
   }
   printf("sum of digits of $original_no is $sum \n");
}
```

```
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ perl B8.SumofDigit.pl 123 456 789
sum of digits of 123 is 6
sum of digits of 456 is 15
sum of digits of 789 is 24
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$
```

B9. Implement an Awk script that folds long lines into 15 columns. Thus, any line that exceeds 15 characters must be broken after 15th and is to be continued with the residue. The inputs to be supplied through a text file created by the user.

B9.Split15.awk

```
{
  st = $0
  while (length(st) > 15) {
      print(substr(st, 1, 15) );
      st = substr(st, 16)
  }
  print(st);
}
```

```
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ cat > long_line.txt
This is a very very long line of text that definitely exceeds fifteen characters for the purpose of demonstrating line folding.
^C
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$ awk -f B9.Split15.awk long_line.txt
This is a very
very long line
of text that de
finitely exceed
s ffifteen char
acters for the
purpose of demo
nstrating line
folding.
shree@shree-INBOOK-Y1-PLUS:~/UNIXLab/Part-B$
```