

Experiment 10:

(a) Write a C program to print the address of a variable and enter a long loop (say using while(1)). Start three to four processes of the same program and observe the printed address values.

Command: nano filename.c

Program: #include<stdio.h>

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

```
#include<unistd.h> Int
```

```
main() {
```

```
printf("Before fork p id :%d\n",getpid());
```

```
int i=0; while(1){ fork(); if(i==3) break;
```

```
i++;
```

```
}
```

```
int var;
```

```
printf("Process id =%d\n Address of var =%u\n",getpid(),&var);
```

```
printf("END\n"); return 0;
```

```
}
```

Terminal: gcc filename.c

./a.out

```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
int main(){
    printf("Berfore fork p id %d\n",getpid());
    int i=0;
    while(1){
        fork();
        if(i==3)
            break;
        i++;
    }
    int var;
    printf("Process id= %d\n Address of var= %u\n",getpid(),&var);
    printf("END\n");
    return 0;
}
```

```

ubuntu@ubuntu2004:~/Desktop/Ex_10$ ./a.out
Before fork p id 8340
Process id= 8340
  Address of var= 76967296
END
Process id= 8343
  Address of var= 76967296
END
ubuntu@ubuntu2004:~/Desktop/Ex_10$ Process id= 8341
  Address of var= 76967296
END
Process id= 8347
  Address of var= 76967296
END
Process id= 8345
  Address of var= 76967296
END
Process id= 8342
  Address of var= 76967296
END
Process id= 8346
  Address of var= 76967296
END
Process id= 8348
  Address of var= 76967296
END
Process id= 8349
  Address of var= 76967296
END
Process id= 8344
Process id= 8350
  Address of var= 76967296
END
  Address of var= 76967296
END
Process id= 8351
  Address of var= 76967296
Process id= 8354
  Address of var= 76967296
END
Process id= 8352
  Address of var= 76967296
END
END
Process id= 8355
Process id= 8353
  Address of var= 76967296
END
  Address of var= 76967296
END

```

(b) Show how Two processes which are members of the relationship parentchild are concurrent from execution point of view, initially the child is copy of the parent, but every process has its own data.

Command: nano filename.c

Program: #include<unistd.h>

#include<sys/types.h>

```
#include<errno.h>

#include<stdio.h>

#include<sys/wait.h>

#include<stdlib.h>

int main()

{

int var=1;

int* p=(int*)malloc(2);

pid_t PID=fork();

*p=0;

if(PID>=0)

{

if(PID==0)

{

printf("\n\nChild Process:\nInitial Value = %d",var);

var=5;

printf("\nNew Value of var =%d",var);

printf("\nAddress of malloc in child =%p",p);

printf("\nAddress of var in child =%p\n",&var);

}
```

```
else  
  
{  
  
printf("\n\nParent process:\nInitial value =%d",var);  
  
var=10;  
  
printf("\nNew value =%d",var);  
  
printf("\nAddress of malloc in parent =%p",p);  
printf("\nAddress of var in child =%p\n",&var);  
  
}  
  
}  
  
return 0;  
  
}
```

```

#include<unistd.h>
#include<sys/types.h>
#include<errno.h>
#include<stdio.h>
#include<sys/wait.h>
#include<stdlib.h>
int main()
{
    int var=1;
    int* p=(int*)malloc(2);
    pid_t PID=fork();
    *p=0;
    if(PID>=0)
    {
        if(PID==0)
        {
            printf("\n\nChild Process:\nInitial Value = %d",var);
            var=5;
            printf("\nNew Value of var = %d",var);
            printf("\nAddress of malloc in child = %p",p);
            printf("\nAddress of var in child = %p\n",&var);
        }
        else
        {
            printf("\n\nParent process:\nInitial value = %d",var);
            var=10;
            printf("\nNew Value = %d",var);
            printf("\nAddress of malloc in parent = %p",p);
            printf("\nAddress of var in child = %p\n",&var);
        }
    }
    return 0;
}

```

```

:/mnt/d$ nano exp10b.c
:/mnt/d$ gcc exp10b.c
:/mnt/d$ ./a.out

```

Parent process:

Child Process:

Initial value = 1

Initial Value = 1

New Value = 10

New Value of var = 5

Address of malloc in parent = 0x7fffe26182a0

Address of malloc in child = 0x7fffe26182a0

Address of var in child = 0x7fffe9df1ec8

Address of var in child = 0x7fffe9df1ec8