Basic Program with fork() System Call

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
int main()
{
  pid_t pid;
  printf("Testign 1\n"); //No output in child because before fork()
  printf("Testign 2\n"); //No output in child because before fork()
  printf("Testign 3\n"); //No output in child because before fork()
  fork();
  printf("Testign of Child\n");
  pid = getpid();
  for(int i = 1; i<=10; i++)
  {
    printf("From Process --> %d, Value = %d\n", pid, i);
  }
  return(0);
}
```

Basic Program with Return Value fork() and getpid() System Call

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
int main()
{
  pid_t pid;
  pid = fork();
  if(pid==0)
  {
    printf("\nOutput from Child -> Hello, I am Child!");
    printf("\nMy Process Id = %d", getpid());
  }
  else
  {
    printf("\nOutput from Parent -> Hello, I am Parent!");
    printf("\nMy Process Id = %d", getpid());
    printf("\nMy Child's Id = %d", pid);
  }
  printf("\nBye Bye from Both!");
  return(0);
}
```

Basic Program for a Zombie Process

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
  pid_t pid;
  pid = fork();
  if(pid==0)
  {
    exit(0);
  }
  else
  {
    sleep(50);
  }
  return(0);
}
```

Basic Program for an Orphan Process

```
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
  pid_t pid;
  pid = fork();
  if(pid==0)
  {
    sleep(10);
    printf("Child Complete");
  }
  else
  {
    printf("Parent Complete");
  }
  return(0);
}
```

Basic Program with wait() System Call

```
#include<sys/wait.h>
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main()
{
  pid_t pid;
  pid = fork();
  if(pid==0)
  {
    for(int i = 1; i<=900000; i++)
    {
      printf("%d-", i);
    }
  }
  else
  {
    wait(NULL);
    printf("\nParent Completed!");
  }
  return(0);
}
```

Basic Program with exec() System Call

```
//Program to be called with exec() system call
#include<stdio.h>
int main() {
  printf("\nI am B!");
  return(0);
}

#include<sys/wait.h>
#include<sys/types.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
int main() {
  pid_t pid;
  pid = fork();
  if(pid==0) {
    printf("\nIn Child");
    execl("./B", "B", NULL);
  }
  Else {
    wait(NULL);
    printf("\nParent Completed!");
  }
}
```