

Assignment 4 Solution

Solution1:

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

main() {

    int pid = fork();

    if (pid < 0) {

        printf("Fork failed.\n");

    } else if (pid == 0) {

        printf("In Child process");

    } else {

        printf("In Parent process");

    }

}
```

Note: Try various cases to explain fork system call.

Solution 2.

```
#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

main () {

    int pid = fork();

    if (pid < 0) {

        printf("Fork failed.\n");

    }

}
```

```

    } else if (pid == 0) {

        printf("Child process, PID: %d\n", getpid());

        printf("Child is exiting.\n");

        exit(0);

    } else {

        printf("Parent process, Child PID: %d\n", pid);

        printf("Parent is waiting for the child to exit...\n");

        wait(NULL);

        printf("Parent's wait is done.\n");

    }

}

```

Solution 3:

Prog1.c

```

#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

int main(int argc, char *argv[])

{

    printf("PID of ex1.c =%d\n", getpid());

    char *args[] = {"hello", "world", NULL} ;

    execv("./prog2.c", args);

    printf("Back to program1");

    return 0;
}

```

```
}
```

Prog2.c

```
#include<stdio.h>

#include<unistd.h>

#include<stdlib.h>

int main(int argc, char *argv[])

{

    Printf(“We are in program 2 \n”);

    Printf(“PID of program 2 is %d\n”, getpid());

    return 0;

}
```

Compile: gcc Prog1.c -o prog1

```
gcc Prog2.c -o prog2
```

```
./prog1
```

Solution 4:

```
#include <stdio.h>
#include <fcntl.h>
int main()
{
    int fd;
    char buffer[80];
    static char message[]=”Hello”;
    fd=open(“myfile.txt”,O_RDWR);
    if (fd != -1)
```

```
{  
printf("myfile.txt opened with read/write access\n");  
write(fd,message,sizeof(message));  
lseek(fd,0,0);  
read(fd,buffer,sizeof(message));  
printf("%s — was written to myfile.txt \n",buffer);  
close(fd);  
}  
}
```