FORK – EXEC – SYSTEM

```c
//======================================================================
// f01.c / JAS
// Fork return value is different for 'parent' and 'child'
// Who is the 'parent' of the 'parent' ? Execute 'ps' command to see ...
//----------------------------------------------------------------------

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
  int fork_return_value;

  printf("before fork...\n");

  fork_return_value=fork();

  printf("I'm process %d: 'fork_return_value'=%d.
         My parent is %d.\n\n",getpid(),fork_return_value,getppid());

  return 0;
}
```

```c
//=======================================================================
// f02.c / JAS
// Father & son. Which one runs first, after fork() ?
// Run several times and interpret results
//-----------------------------------------------------------------------

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
  int pid;

  printf("before fork...\n"); // remove '\n' and see what happens

  pid=fork();

  if (pid > 0)
    printf("I'm the parent (PID=%d)\n\n",getpid());
  else
    printf("I'm the son (PID=%d)\n\n",getpid());
  printf ("PID=%d exiting ...\n",getpid());

  return 0;
}


//=======================================================================
// f02a.c / JAS
// Fork & output buffering
// Equal to f02.c with '\n' remove in "before fork ..." message
//-----------------------------------------------------------------------

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
  int pid;

  printf("before fork..."); // '\n' was removed

  pid=fork();

  if (pid > 0)
    printf("I'm the parent (PID=%d)\n\n",getpid());
  else
    printf("I'm the son (PID=%d)\n\n",getpid());
  printf ("PID=%d exiting ...\n",getpid());

  return 0;
}
```

```c
//========================================================================
// f03.c / JAS
// Fork & output buffering
// Equal to f02.c print("before fork ...") replaced by write(...)
//------------------------------------------------------------------------

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
  int pid;

  write(STDOUT_FILENO,"before fork...",14); // printf() replaced by write()

  pid=fork();

  if (pid > 0)
    printf("I'm the parent (PID=%d)\n\n",getpid());
  else
    printf("I'm the son (PID=%d)\n\n",getpid());
  printf ("PID=%d exiting ...\n",getpid());

  return 0;
}
```

```c
//========================================================================
// f04.c / JAS
// Basic synchronization. Father waits for the son to end.
//------------------------------------------------------------------------

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

int main(void)
{
  pid_t pid, pidSon;
  int status;

  pid=fork();
  if (pid > 0) {
    pidSon = wait(&status);
    printf("I'm the parent (PID=%d)\n\n",getpid());
    printf("My son %d exited with exit code %d\n",
            pidSon, WEXITSTATUS(status));}
  else
  {
    printf("I'm the son (PID=%d)\n\n",getpid());
    exit( getpid() % 10 );
  }
  printf ("PID=%d exiting ...\n",getpid());
  return 0;
}
```

```c
//========================================================================
// f05.c / JAS
// zombie's
// In another terminal, execute command 'ps u'
//------------------------------------------------------------------------

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
  int pid;

  pid=fork();
  if (pid > 0) {
    printf("I'm the parent (PID=%d)\n\n",getpid());
    sleep(10); }
  else {
    printf("I'm the son (PID=%d)\n\n",getpid());
  }
  printf ("PID=%d exiting ...\n",getpid());
  return 0;
}
```

```c
//======================================================================
// f06.c / JAS
// Tree of child processes with some zombies
// In another terminal, execute command 'ps u'
//----------------------------------------------------------------------

#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main(void)
{
 int i, pid;

 for (i=1;i<=3;i++) {
  pid=fork();
  if (pid > 0) {
   printf("I'm the parent (PID=%d)\n\n",getpid());
   sleep(5);
  }
  else {
   printf("I'm the son (PID=%d). My parent is %d\n\n",getpid(),getppid());
   break;  // NOTE THIS
  }
 }
 printf ("PID=%d exiting ...\n",getpid());
 return 0;
}
```

```c
//======================================================================
// e01.c / JAS
// execl() & execlp()
//----------------------------------------------------------------------

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

int main(void)
{
  int pid;

  pid=fork();
  if (pid > 0) {
    wait(NULL); //father does not care w/exit status of the son ...
    printf("I'm the parent (PID=%d)\n\n",getpid()); }
  else {
    printf("I'm the son (PID=%d)\n\n",getpid());
    execl("ls","ls","-la",NULL); //try with execlp()
    printf(".... \n"); //which message makes sense, here ?
  }
  printf ("PID=%d exiting ...\n",getpid());
  return 0;
}
```

```c
//========================================================================
// e01.c / JAS
// execl() & execlp()
//------------------------------------------------------------------------

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

int main(void)
{
  int pid;
  int status;

  pid=fork();
  if (pid > 0) {
    wait(&status);
    printf("I'm the parent (PID=%d)\n\n",getpid());
    printf("My son exited with EXIT CODE = %d\n",WEXITSTATUS(status));}
  else {
    printf("I'm the son (PID=%d)\n\n",getpid());
    execlp("ls","ls","-la",NULL); //try with execl()
    //execl("./e01_aux","e01_aux","3",NULL);
    printf(".... \n"); //which message makes sense, here ?
  }
  printf ("PID=%d exiting ...\n",getpid());
  return 0;
}



//========================================================================
// e01_aux.c / JAS
// To be executed with e01.c
//------------------------------------------------------------------------

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
  int i, n;

  n = atoi(argv[1]);
  for (i=1; i<=n; i++)
  {
    printf("CHILD  (%d - %d): Hello father ...%d!\n",getpid(),getppid(),i);
  }

  return 10;
}
```

```c
//==========================================================================
// e02.c / JAS
// exec()
//--------------------------------------------------------------------------

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

int main(void)
{
  int pid;
  int status;

  pid=fork();
  if (pid > 0) {
    wait(&status);
    printf("I'm the parent (PID=%d)\n\n",getpid());
    printf("My son exited with EXIT CODE = %d\n",WEXITSTATUS(status));}
  else {
    printf("I'm the son (PID=%d)\n\n",getpid());
    execlp("cat","cat","e02.c",NULL); // change "e02.c" to "xxxxx.c"
    printf("exec() failed !!! \n");
    exit(1);
  }
  printf ("PID=%d exiting ...\n",getpid());
  return 0;
}
```

```c
//======================================================================
// e03.c / JAS
// exec()
//----------------------------------------------------------------------

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

int main(void)
{
  int pid;
  int status;

  pid=fork();
  if (pid > 0) {
    wait(&status);
    printf("I'm the parent (PID=%d)\n\n",getpid());
    printf("My son exited with EXIT CODE = %d\n",WEXITSTATUS(status));}
  else {
    printf("I'm the son (PID=%d)\n\n",getpid());
    execlp("cat","cat","e03.c",">","e03_copy.c",NULL);
    // note the "no such file or directory" errors of "cat"...!
    printf("exec() failed !!! \n");
    exit(1);
  }
  printf ("PID=%d exiting ...\n",getpid());
  return 0;
}
```

```c
//======================================================================
// e04.c / JAS
// exec()
//----------------------------------------------------------------------

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


int main(void)
{
  int pid;
  int status;
  char *arg[]={"ls","-laR",NULL};

  printf("before fork\n");
  pid=fork();
  if (pid > 0) {
    wait(&status);
    printf("I'm the parent (PID=%d)\n\n",getpid()); }
  else {
    printf("I'm the son (PID=%d)\n\n",getpid());
    execvp("ls",arg);
    printf("EXEC failed\n");
  }
  printf ("PID=%d exiting ...\n",getpid());
  return 0;
}
```

```c
//========================================================================
// e05.c / JAS
// A simple command interpreter
//------------------------------------------------------------------------

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

int main(void)
{
  int pid, pid_terminated, status;
  char cmd[100];

  printf("Command (OR quit)? "); scanf("%s",cmd);

  while (strcmp(cmd,"quit") != 0)
  {
    pid=fork();

    if (pid>0)
    { // COMMENT THE 2 LINES BELOW TO SEE THE ZOMBIES
      pid_terminated = wait(&status);
      printf("PARENT: son %d terminated with exit code %d\n",
        pid_terminated,WEXITSTATUS(status));
    }
    else
    {
      execlp(cmd,cmd,NULL);
      printf("Command not found !!!\n");
      exit(1);
    }

    printf("Command? "); scanf("%s",cmd);
  }

  return 0;
}
```

```c
//=======================================================================
// s01.c / JAS
// system()
//-----------------------------------------------------------------------

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

int main(void)
{
  int pid;

  printf("before fork\n");
  pid=fork();
  if (pid > 0) {
    wait(NULL);
    printf("I'm the parent (PID=%d)\n\n",getpid()); }
  else {
    printf("I'm the son (PID=%d)\n\n",getpid());
    system("ls /usr/include/s*.h -la"); //NOTE: system() "expands" s*.h
    // try also system("cat s01.c > s01_copy.c");
    printf("\n AFTER system() call\n"); //WHY NOT FAILED, in this case,
like in exec()
    exit(0);
  }
  printf ("PID=%d exiting ...\n",getpid());
  return 0;
}
```