CSI 402 – Systems Programming – Handout 14.1
Use of `fork`, `wait` and `exec`: Additional Example I

Note: In the following example, the parent forks a child and waits for the child to complete the program given by the executable `parg`. The child process prints its pid, executes the `parg` program and then exits. The `parg` program (whose source code is given on the back of this page) merely prints each of its command line arguments except `argv[0]`. When the child is finished, the parent process prints the pid and the exit status of the child and then exits. When you run this program, the output produced by the child is printed before that of the parent.

Source code for the parent process:

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

int main(void) {
  pid_t  child;    /* Child pid returned by fork. */
  int    cstatus;  /* Exit status of child. */
  pid_t  c;        /* Pid of child to be returned by wait. */

  if ((child = fork()) == 0) {
     /* Child process.  To begin with, it prints its pid. */
     printf("Child: PID of Child = %ld\n", (long) getpid());

     /* Child will now execute the parg program. */
     execlp("parg", "parg", "Another", "Example", "for", "fork-wait-exec", NULL);

     /* If the child process reaches this point, then execlp must have failed. */
     fprintf(stderr, "Child process could not do execlp.\n"); exit(1);
  }

  else { /* Parent process. */
     if (child == (pid_t)(-1)) {
        fprintf(stderr, "Fork failed.\n"); exit(1);
     }
     else {
        c = wait(&cstatus); /* Wait for child to complete. */
        printf("Parent: Child  %ld exited with status = %d\n", (long) c, cstatus);
     }
  }
  return  0;
} /* End of main. */
```

(over)

Note: The following C program must be compiled and linked to produce an executable called `parg`. This executable must be in a directory that is included in the value of the `PATH` environment variable. This program is executed by the child process after printing its pid.

Source code for the parg program:

```
#include  <stdio.h>

/* This program will be executed by the child process; the name of the */
/* executable must be parg.                                             */

/* This program merely prints its arguments, except the name of the */
/* program (i.e., argv[0]) itself.                                   */

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

  int i;  /* Loop index. */

  /* In the following loop, i starts at 1 since we don't want to */
  /* print argv[0].                                               */

  for (i = 1; i < argc; i++)
     printf("%s\n", argv[i]);

  return 0;

} /* End of main. */
```

Output: (Note that the pid of the child will vary from run to run.)

```
Child: PID of Child = 15597
Another
Example
for
fork-wait-exec
Parent: Child  15597 exited with status = 0
```