CSC4421 Computer Operating System

Lab 04

exec()

- Replaces current process image with new process image.
- Does NOT create a new process.
- Returns -1 on failure.
- Nothing returned if successful.

exec()

```
EXEC(3)                    Linux Programmer's Manual                    EXEC(3)

NAME
       execl, execlp, execle, execv, execvp, execvpe - execute a file

SYNOPSIS
       #include <unistd.h>

       extern char **environ;

       int execl(const char *path, const char *arg, ...
                       /* (char  *) NULL */);
       int execlp(const char *file, const char *arg, ...
                       /* (char  *) NULL */);
       int execle(const char *path, const char *arg, ...
                       /*, (char *) NULL, char * const envp[] */);
       int execv(const char *path, char *const argv[]);
       int execvp(const char *file, char *const argv[]);
       int execvpe(const char *file, char *const argv[],
                       char *const envp[]);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

 Manual page exec(3) line 1 (press h for help or q to quit)
```

execl()

- 1st argument: path name of executable
- Other arguments: command line arguments, ending with NULL.
- Example: execl("/bin/ls", "ls", "-a", "-l", NULL)

execv()

- 1st argument: path name of executable
- 2nd argument: NULL terminated array of arguments
  - First argument of array can be the command or " "
- Example:

```
static char* args[ ] = {"ls", "-a", "-l", NULL};
execv("/bin/ls", args);
```

execlp()

- execl, but not required to specify full path to the command
- 1st argument: (path) name of executable
- Other arguments: command line arguments, ending with NULL.
- Example: execl("/ls", "ls", "-a", "-l", NULL)

execvp()

- execv, but not required to specify full path to the command
- 1st argument: (path) name of executable
- 2nd argument: NULL terminated array of arguments
  - First argument of array can be the command or " "
- Example:

```
static char* args[ ] = {"ls", "-a", "-l", NULL};
execv("/ls", args);
```

strtok()

```
STRTOK(3)                  Linux Programmer's Manual                  STRTOK(3)

NAME
       strtok, strtok_r - extract tokens from strings

SYNOPSIS
       #include <string.h>

       char *strtok(char *str, const char *delim);

       char *strtok_r(char *str, const char *delim, char **saveptr);

   Feature Test Macro Requirements for glibc (see feature_test_macros(7)):

       strtok_r(): _SVID_SOURCE || _BSD_SOURCE || _POSIX_C_SOURCE >= 1 ||
       _XOPEN_SOURCE || _POSIX_SOURCE

DESCRIPTION
       The strtok() function breaks a string into a sequence of zero  or  more
       nonempty tokens.  On the first call to strtok() the string to be parsed
       should be specified in str.  In each subsequent call that should  parse
       the same string, str must be NULL.

 Manual page strtok(3) line 1 (press h for help or q to quit)
```

strtok()

- Break a string into tokens.
- char *strtok(char *str, const char *delim)
- 1st call to strtok: pass string to str
- Calls 2 -> n: pass NULL to str
- 2nd – (n-1) calls: tokens returned
- Nth call: NULL is returned

strtok()

- char input[] = "Hello World!"; //a string in c is terminated with " \n“ in the character array
- strtok(input, " "); //returns “Hello”
- strtok(NULL, " "); //returns “World!”
- strtok(NULL, " "); // returns NULL