SISTEMAS OPERATIVOS

Tecnológico de Monterrey, Campus Querétaro

Laboratorio 6 - Prcesos

1. Introducción

En este laboratorio, no encontraremos trabajando con la administración de procesos en UNIX.

2. Ejercicio 1

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

int main(int argc, char* argv[]) {
  pid_t pid, pid1;

  /* fork a child process */
  pid = fork();

  if (pid < 0) { /* error ocurred */
    perror(argv[0]);
    return -1;
  } else if (pid == 0) { /* child process */
    pid1 = getpid();
    printf("child - pid = %d\n", pid);  /* A */
    printf("child - pid = %d\n", pid1); /* B */
    return 0;
  } else { /* parent process */
    pid1 = getpid();
    printf("child - pid = %d\n", pid);  /* C */
    printf("child - pid = %d\n", pid1); /* D */
    /* parent will wait for the child to complete */
    wait(NULL);
  }
  return 0;
}
```

1. TIdentifica cuáles son los valores de pid de las líneas `A`, `B`, `C` y `D`.

2. Explica el porqué del resultado.

3. Ejercicio 2

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

```c
#include <sys/wait.h>
#include <sys/types.h>

int value = 5;

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

  /* fork a child process */
  pid = fork();

  if (pid < 0) { /* error ocurred */
    perror(argv[0]);
    return -1;
  } else if (pid == 0) { /* child process */
    value += 15;
    return 0;
  } else { /* parent process */
    /* parent will wait for the child to complete */
    wait(NULL);
    printf("PARENT: value = %d\n", value); /* LINEA A */
  }
  return 0;
}
```

1. Tomando como base el código anterior, ¿qué valor despliega el programa (`LINEA A`)?
2. Explica el porqué del resultado.

4. Ejercicio 3

4.1. Programa 1

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

#define NUM   5
#define SLEEP 3

void child_process() {
  sleep(SLEEP);
  printf("PID = %i PPID = %i\n", getpid(), getppid());
  exit(0);
}

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

  for (i = 0; i < NUM; i++) {
    /* fork a child process */
    pid = fork();
```

```
    if (pid < 0) { /* error ocurred */
      perror(argv[0]);
      return -1;
    } else if (pid == 0) { /* child process */
      child_process();
      return 0;
    } else { /* parent process */
      wait(NULL);
      printf("PARENT PID: %d\n", getpid());
    }
  }
  return 0;
}
```

4.2. Programa 2

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

#define NUM   5
#define SLEEP 3

void child_process() {
  sleep(SLEEP);
  printf("PID = %i PPID = %i\n", getpid(), getppid());
  exit(0);
}

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

  for (i = 0; i < NUM; i++) {
    /* fork a child process */
    pid = fork();

    if (pid < 0) { /* error ocurred */
      perror(argv[0]);
      return -1;
    } else if (pid == 0) { /* child process */
      child_process();
      return 0;
    } else { /* parent process */
      /* do nothing */
    }
  }

  sleep(SLEEP);
  int rid;
  while (i > 0) {
    rid = wait(NULL);
    printf("The process %i has ended\n", rid);
    i--;
  }
  return 0;
```

```
}
```

1. Ejecuta ambos programas.
2. Explica cuál es la diferencia entre ambos programas. ¿Cómo se crean los procesos?