



INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO



SISTEMAS OPERATIVOS

CORTÉS GALICIA JORGE

MENDOZA PARRA SERGIO

OLVERA AGUILA LEONARDO DANIEL

PAZ SÁNCHEZ BRANDON

2CM7

**PRÁCTICA 5. ADMINISTRADOR DE PROCESOS EN LINUX Y
WINDOWS (2)**

LUNES, 13 DE NOVIEMBRE DE 2017

Contenido

Competencias 3

Desarrollo 4

Análisis critico..... 7

Conclusiones..... 7

Competencias

El alumno aprende a familiarizarse con el administrador de procesos del sistema operativo Linux y Windows a través de la creación de nuevos procesos por copia exacta de código y/o por sustitución de código para el desarrollo de aplicaciones concurrentes sencillas.

Desarrollo

Sección Linux

1. A través de la ayuda en línea que proporciona Linux investigue el funcionamiento de las funciones **pthread_create()**, **pthread_join()**, **pthread_exit()**, **scandir()**, **stat()**. Explique los argumentos y retorno de cada función.

pthread_create(): La función pthread_create() inicia un nuevo hilo en el proceso de llamada. El nuevo subproceso inicia su ejecución mediante la invocación de start_routine(); arg se pasa como único argumento de start_routine(). El argumento attr apunta a la estructura pthread_attr_t cuyo contenido se utiliza en el momento de la creación de hilo para determinar atributos para el nuevo hilo; Esta estructura se inicializa utilizando pthread_attr_init y funciones relacionadas. Si los atributos son NULL, el hilo se crea con atributos predeterminados.

```
#include <pthread.h>
```

```
int pthread_create(pthread_t *thread, const pthread_attr_t  
*attr,  
void *(*start_routine) (void *), void *arg);
```

Compile and link with `-pthread`.

pthread_join(): La función pthread_join() se espera para el subproceso especificado por hilo para terminar. Si ese hilo ya ha terminado, entonces pthread_join() devuelve un valor inmediatamente. El subproceso especificado por hilo debe ser acoplable.

Si retval no es NULL, entonces pthread_join() copia el estado de salida de la rosca del objetivo (es decir, el valor que el subproceso de destino suministra a pthread_exit en el lugar apuntado por * retval. Si el subproceso de destino fue cancelado, entonces

PTHREAD_CANCELED se coloca en * retval.

```
#include <pthread.h>
```

```
int pthread_join(pthread_t thread, void ** value_ptr);
```

pthread_self(): La función pthread_self() devuelve el identificador de subproceso de llamada. Este es el mismo valor devuelto en * hilo de rosca en la llamada pthread_create que creó este hilo.

```
#include <pthread.h>
```

```
pthread_t pthread_self(void)
```

pthread_exit(): La función de pthread_exit() termina el subproceso de llamada y devuelve un valor mediante retval que (si el hilo es unible) está disponible para otro hilo en el mismo proceso que llama a pthread_join.

```
#include <pthread.h>
```

```
void pthread_exit (void *retval);
```

scandir(): La función scandir() escanea el directorio dirp, llamando a filter() a cada entrada de directorio. Las entradas para el que el filter() devuelve distinto de cero son almacenadas en cadenas asignadas a través de malloc, ordenadas utilizando qsort con la función de comparación compar(), y recogiendo en orden la lista de nombres que se asignan a través de malloc. Si el filtro es NULL, todas las entradas están seleccionadas.

```
#include <dirent.h>
```

```
int scandir(const char *dirp, struct dirent ***namelist,  
            int (*filter)(const struct dirent *),  
            int (*compar)(const struct dirent **, const struct dirent **));
```

stat(): Estas funciones devuelven información acerca de un archivo, en el búfer apuntada por buf. No hay permisos necesarios para el propio archivo, pero en el caso de stat(), fstatat(), y lstat() - ejecución (búsqueda) se requiere el permiso de todos los directorios en la ruta de acceso que conducen al archivo.

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

```
#include <sys/stat.h>
```

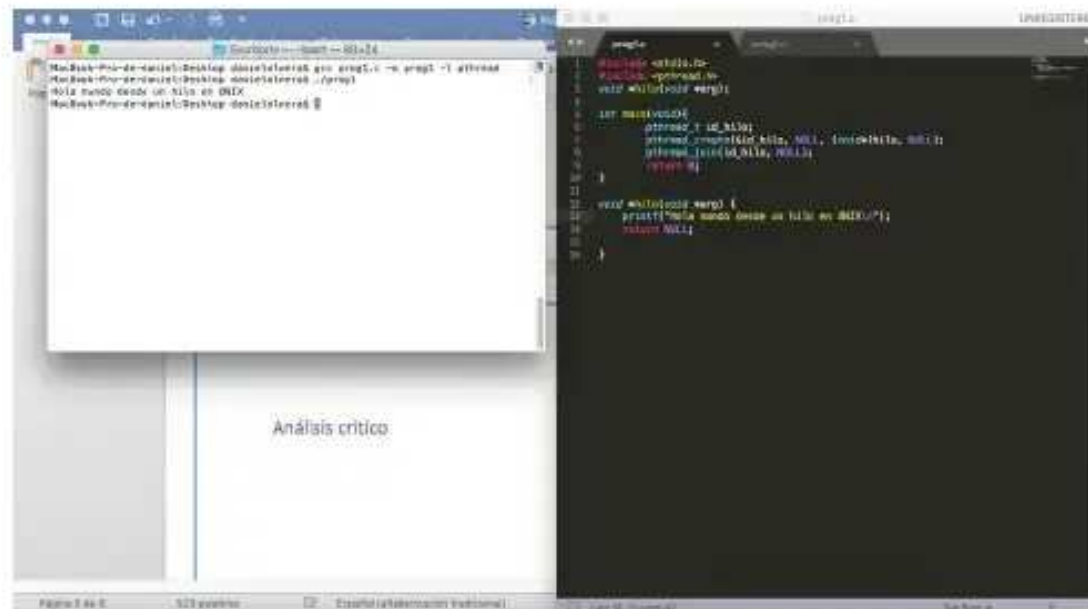
```
#include <unistd.h>
```

```
int stat(const char *pathname, struct stat *buf);
```

```
int fstat(int fd, struct stat *buf);
```

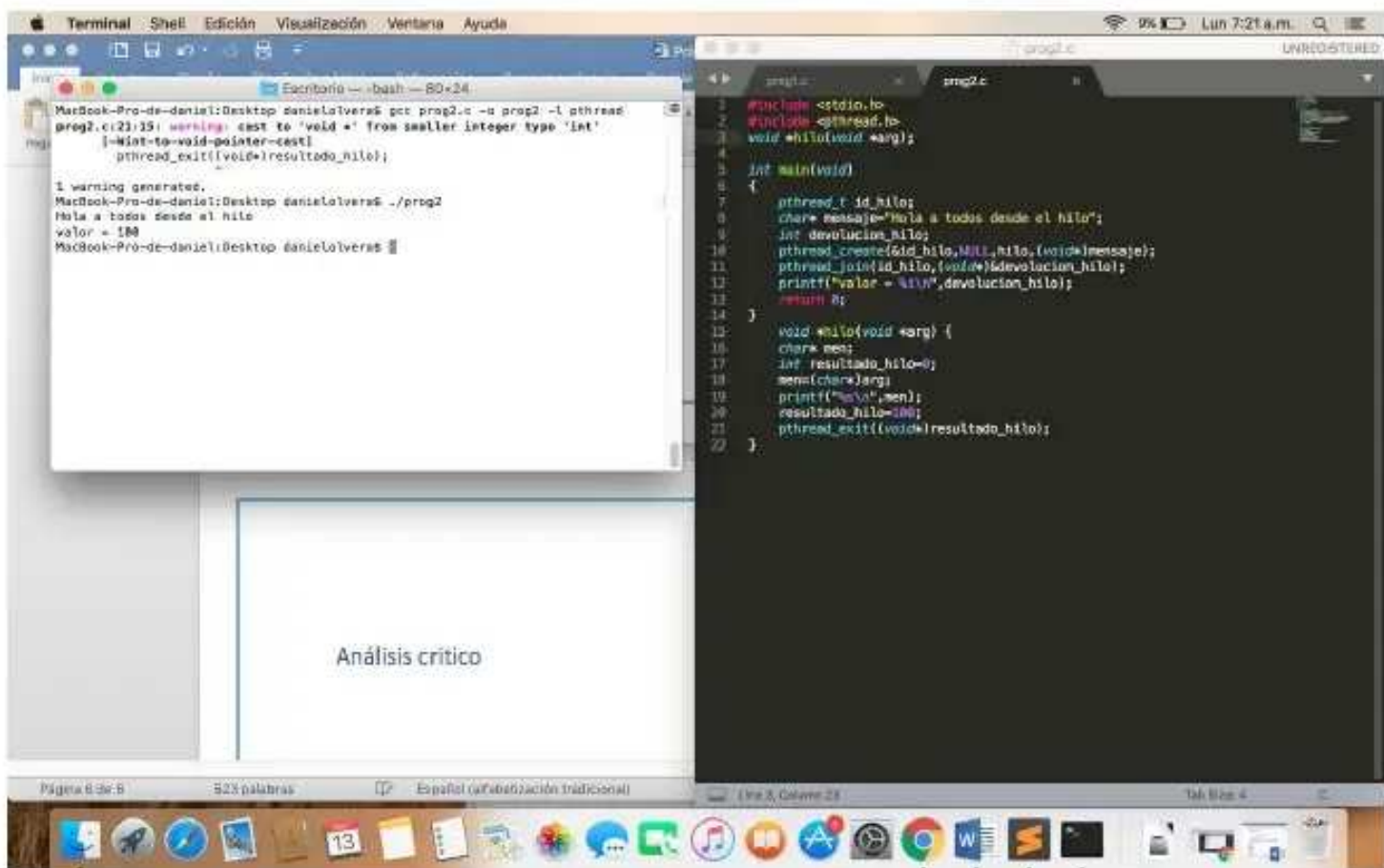
```
int lstat(const char *pathname, struct stat *buf);
```

2. Capture, compile y ejecute el programa de creación de un nuevo hilo en Linux. Observe su funcionamiento.



```
MacBook-Pro-de-daniel:Desktop danieloliveras$ gcc prog1.c -o prog1 -l pthread
MacBook-Pro-de-daniel:Desktop danieloliveras$ ./prog1
Hola mundo desde un hilo en OSX
MacBook-Pro-de-daniel:Desktop danieloliveras$
```

```
1 #include <stdio.h>
2 #include <pthread.h>
3 void *hilo(void *arg);
4
5 int main(void)
6 {
7     pthread_t id_hilo;
8     pthread_create(&id_hilo, NULL, (void*)hilo, NULL);
9     pthread_join(id_hilo, NULL);
10    return 0;
11 }
12
13 void *hilo(void *arg) {
14     printf("Hola mundo desde un hilo en OSX\n");
15     return NULL;
16 }
```



```
MacBook-Pro-de-daniel:Desktop danieloliveras$ gcc prog2.c -o prog2 -l pthread
prog2.c:21:15: warning: cast to 'void *' from smaller integer type 'int'
[-Wint-to-void-pointer-cast]
    pthread_exit((void*)resultado_hilo);
                   ^
1 warning generated.
MacBook-Pro-de-daniel:Desktop danieloliveras$ ./prog2
Hola a todos desde el hilo
valor = 100
MacBook-Pro-de-daniel:Desktop danieloliveras$
```

```
1 #include <stdio.h>
2 #include <pthread.h>
3 void *hilo(void *arg);
4
5 int main(void)
6 {
7     pthread_t id_hilo;
8     char *mensaje = "Hola a todos desde el hilo";
9     int devolucion_hilo;
10    pthread_create(&id_hilo, NULL, (void*)hilo, (void*)mensaje);
11    pthread_join(id_hilo, (void*)&devolucion_hilo);
12    printf("valor = %i\n", devolucion_hilo);
13    return 0;
14 }
15
16 void *hilo(void *arg) {
17     char *men;
18     int resultado_hilo = 0;
19     memcpy(&men, arg, sizeof(char*));
20    printf("Hola\n", men);
21    resultado_hilo = 100;
22    pthread_exit((void*)&resultado_hilo);
23 }
```

Análisis crítico

Conclusiones