Primera práctica de Sistemas Operativos

Generado por Doxygen 1.8.14

Índice general

Capítulo 1

Índice de clases

1.1. Lista de clases

Lista de las clases, estructuras, uniones e interfaces con una breve descripción:

Capítulo 2

Indice de archivos

2.1. Lista de archivos

Lista de todos los archivos documentados y con descripciones breves:

Capítulo 3

Documentación de las clases

3.1. Referencia de la Estructura _estructura

estructura generada al principio de la ejecución

Atributos públicos

- char * cadena
- int * entero

3.1.1. Descripción detallada

estructura generada al principio de la ejecución

Esta estructura almacena una cadena de 100 caracteres y un entero. Se usa para comparar el uso de memoria entre los hilos y los procesos

3.1.2. Documentación de los datos miembro

3.1.2.1. cadena

```
char * _estructura::cadena
```

Cadena de 100 caracteres

3.1.2.2. entero

```
int * _estructura::entero
```

Puntero a un entero

La documentación para esta estructura fue generada a partir de los siguientes ficheros:

- ejercicio12a.c
- ejercicio12b.c

3.2. Referencia de la Estructura param

Estructura de parámetros de un hilo.

Atributos públicos

- int ** matriz
- int mult
- int dim
- int id

3.2.1. Descripción detallada

Estructura de parámetros de un hilo.

Esta estructura almacena la información necesaria para los hilos en este ejercicio

3.2.2. Documentación de los datos miembro

3.2.2.1. dim

```
int param::dim
```

Dimension de las matrices

3.2.2.2. id

```
int param::id
```

Identificador del hilo

3.2.2.3. matriz

```
int** param::matriz
```

Matriz

3.2.2.4. mult

```
int param::mult
```

Multiplicador

La documentación para esta estructura fue generada a partir del siguiente fichero:

- ejercicio13.c

Capítulo 4

Documentación de archivos

4.1. Referencia del Archivo ejercicio12a.c

Ejercicio para la comparación de tiempo: procesos.

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

Clases

- struct _estructura

  estructura generada al principio de la ejecución

defines

- #define NUMERO_PROCESOS 100

typedefs

- typedef struct _estructura estructura

  estructura generada al principio de la ejecución

Funciones

- void calcula_primos (int N)

  calcula los n primos numeros primos
- int main (int argc, char **argv)

  Función principal del programa.

4.1.1. Descripción detallada

Ejercicio para la comparación de tiempo: procesos.

Este fichero, en conjunto con el ejercicio12b.c, sirven para comparar la eficiencia de los threads comparados con los procesos

Autor

Miguel Arconada Manteca y José Manuel Chacón Aguilera

Fecha

8-3-2018

4.1.2. Documentación de los 'defines'

4.1.2.1. NUMERO_PROCESOS

```
#define NUMERO_PROCESOS 100
```

Numero total de procesos que queremos crear

4.1.3. Documentación de los 'typedefs'

4.1.3.1. estructura

```
typedef struct _estructura estructura
```

estructura generada al principio de la ejecución

Esta estructura almacena una cadena de 100 caracteres y un entero. Se usa para comparar el uso de memoria entre los hilos y los procesos

4.1.4. Documentación de las funciones

4.1.4.1. calcula_primos()

```
void calcula_primos (
            int N )
```

calcula los n primos numeros primos

Esta funcion calcula los n primero numeros primos

Parámetros

N	numero de primos a calcular

Devuelve

void

4.1.4.2. main()

```
int main (
            int argc,
            char ** argv )
```

Función principal del programa.

Este programa lanza 100 procesos simultáneamente, en los que se calculan los N primeros primos

Parámetros

argc	numero de parametros de entrada
argv	array de cada parametro. El primero es el nombre del programa, y el segundo es N

Devuelve

0 si todo se ejecuta correctamente, y -1 en cualquier otro caso

4.2. Referencia del Archivo ejercicio12b.c

Ejercicio para la comparación de tiempo: hilos.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <pthread.h>
#include <time.h>
#include <math.h>
```

Clases

- struct _estructura

  estructura generada al principio de la ejecución

defines

- #define NUMERO_HILOS 100

typedefs

- typedef struct _estructura estructura

  estructura generada al principio de la ejecución

Funciones

- void * calcula_primos (void *arg)

  calcula los N primeros primos
- int main (int argc, char **argv)

  Función principal del programa.

4.2.1. Descripción detallada

Ejercicio para la comparación de tiempo: hilos.

Este fichero, en conjunto con el ejercicio12b.c, sirven para comparar la eficiencia de los threads comparados con los procesos

Autor

Miguel Arconada Manteca y José Manuel Chacón Aguilera

Fecha

8-3-2018

4.2.2. Documentación de los 'defines'

4.2.2.1. NUMERO_HILOS

```
#define NUMERO_HILOS 100
```

Numero total de hilos que queremos crear

4.2.3. Documentación de los 'typedefs'

4.2.3.1. estructura

```
typedef struct _estructura estructura
```

estructura generada al principio de la ejecución

Esta estructura almacena una cadena de 100 caracteres y un entero. Se usa para comparar el uso de memoria entre los hilos y los procesos

4.2.4. Documentación de las funciones

4.2.4.1. calcula_primos()

```
void * calcula_primos (
            void * arg )
```

calcula los N primeros primos

calcula_primos calcula los N primeros números primos

Parámetros

arg	puntero a un entero que contiene N hecho casting a void*

Devuelve

void

4.2.4.2. main()

```
int main (
            int argc,
            char ** argv )
```

Función principal del programa.

Este programa lanza 100 hilos simultáneamente, en los que se calculan los N primeros primos

Parámetros

argc	numero de parametros de entrada
argv	array de cada parametro. El primero es el nombre del programa, y el segundo es N

Devuelve

0 si todo se ejecuta correctamente, y -1 en cualquier otro caso

4.3. Referencia del Archivo ejercicio13.c

Ejercicio sobre paso de parámetros en funciones.

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

Clases

- struct param

  Estructura de parámetros de un hilo.

defines

- #define BUFFER_SIZE 256
- #define ARRAY_SIZE 30

typedefs

- typedef struct param Param

  Estructura de parámetros de un hilo.

Funciones

- void ∗ calcula_matriz (void ∗arg)

  multiplica una matriz por un escalar
- int main ()

  Función principal del programa.

4.3.1. Descripción detallada

Ejercicio sobre paso de parámetros en funciones.

Este fichero contiene nuestra solución al ejercicio 13 enunciado en el pdf de la práctica

Autor

Miguel Arconada Manteca y José Manuel Chacón Aguilera

Fecha

8-3-2018

4.3.2. Documentación de los 'defines'

4.3.2.1. ARRAY_SIZE

```
#define ARRAY_SIZE 30
```

Tamanio maximo de los buffers valores

4.3.2.2. BUFFER_SIZE

```
#define BUFFER_SIZE 256
```

Tamanio maximo del buffer cadena_aux

4.3.3. Documentación de los 'typedefs'

4.3.3.1. Param

```
typedef struct param Param
```

Estructura de parámetros de un hilo.

Esta estructura almacena la información necesaria para los hilos en este ejercicio

4.3.4. Documentación de las funciones

4.3.4.1. calcula_matriz()

```
void * calcula_matriz (
            void * arg )
```

multiplica una matriz por un escalar

calcula_matriz multiplica una matriz por un escalar. Está almacenada en forma de puntero a función para ser ejecutada por varios threads simultáneamente.

Parámetros

arg	estructura del tipo Param que contiene la matriz, el escalar, la dimension y un identificador del hilo

Devuelve

void

4.3.4.2. main()

```
int main (
            void  )
```

Función principal del programa.

Este programa pide por consola una dimensión, dos multiplicadores y dos matrices, y lanza dos hilos que multiplican cada matriz por los escalares, y que van imprimiendo cada línea según la calculan

Parámetros

void	

Devuelve

0 si todo se ejecuta correctamente, y -1 en cualquier otro caso

4.4. Referencia del Archivo ejercicio4a.c

Ejercicio 4a de la Práctica.

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

defines

- #define NUM_PROC 6

Funciones

- int **main** (void)

4.4.1. Descripción detallada

Ejercicio 4a de la Práctica.

En este ejercicio vemos el ejemplo de ejecucioón de un programa que lanza procesos hijos.

Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

Fecha

8-3-2018

4.4.2. Documentación de los 'defines'

4.4.2.1. NUM_PROC

```
#define NUM_PROC 6
```

Número de iteraciones del bucle creador de procesos

4.5. Referencia del Archivo ejercicio4b.c

Ejercicio 4b de la Práctica.

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

defines

- #define NUM_PROC 6

Funciones

- int **main** (void)

4.5.1. Descripción detallada

Ejercicio 4b de la Práctica.

En este ejercicio vemos el ejemplo de ejecucioón de un programa que lanza procesos hijos.

Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

Fecha

8-3-2018

4.5.2. Documentación de los 'defines'

4.5.2.1. NUM_PROC

```
#define NUM_PROC 6
```

Número de iteraciones del bucle creador de procesos

4.6. Referencia del Archivo ejercicio5a.c

Ejercicio 5a de la Práctica.

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

defines

- #define NUM_PROC 6

Funciones

- int **main** (void)

4.6.1. Descripción detallada

Ejercicio 5a de la Práctica.

En este ejercicio vemos el ejemplo de ejecucioón de un programa que lanza procesos hijos.

Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

Fecha

8-3-2018

4.6.2. Documentación de los 'defines'

4.6.2.1. NUM_PROC

```
#define NUM_PROC 6
```

Número de iteraciones del bucle creador de procesos

4.7. Referencia del Archivo ejercicio5b.c

Ejercicio 5b de la Práctica.

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

defines

- #define NUM_PROC 6

Funciones

- int **main** (void)

4.7.1. Descripción detallada

Ejercicio 5b de la Práctica.

En este ejercicio vemos el ejemplo de ejecucioón de un programa que lanza procesos hijos.

Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

Fecha

8-3-2018

4.7.2. Documentación de los 'defines'

4.7.2.1. NUM_PROC

```
#define NUM_PROC 6
```

Número de iteraciones del bucle creador de procesos

4.8. Referencia del Archivo ejercicio6.c

Ejercicio 6 de la Práctica.

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

defines

- #define MAX_CAD 128

Funciones

- int **main** ()

4.8.1. Descripción detallada

Ejercicio 6 de la Práctica.

En este ejercicio combinamos cuestiones teóricas sobre reserva de memoria dinamica y ejecución del programa en procesos distintos

Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

Fecha

8-3-2018

4.8.2. Documentación de los 'defines'

4.8.2.1. MAX_CAD

```
#define MAX_CAD 128
```

Tamaño máximo de las cadenas inicializadas

4.9. Referencia del Archivo ejercicio8.c

Ejercicio 8 de la Práctica.

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

defines

- #define MAX_PATH 256

Funciones

- int **main** (int argc, char **argv)

4.9.1. Descripción detallada

Ejercicio 8 de la Práctica.

En este ejercicio lanzamos tantos procesos hijo como escpecificados en los parámetros de entrada de la función main y además hacemos que estos procesos hijos ejecuten un código distinto al de este programa todo esto lo conseguimos con una combinación de las rutinas fork() y la familia de rutinas exec().

Autor

José Manuel Chacón Aguilera y Miguel Arconada Manteca

Fecha

8-3-2018

4.9.2. Documentación de los 'defines'

4.9.2.1. MAX_PATH

```
#define MAX_PATH 256
```

Tamaño máximo de la ruta

4.10. Referencia del Archivo ejercicio9.c

Ejercicio para la comunicacion entre procesos mediante tuberias.

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

defines

- #define LECTURA 0
- #define ESCRITURA 1
- #define BUFFER_SIZE 512
- #define MENSAJE_SIZE 512

Funciones

- int factorial (int n)

  calcula el factorial de un numero
- int main (int argc, char **argv)

  Función principal del programa.

4.10.1. Descripción detallada

Ejercicio para la comunicacion entre procesos mediante tuberias.

En este fichero hacemos uso de las pipes para implementar la comunicacion entre un proceso padre y cuatro procesos hijos, a los cuales les manda dos enteros, y estos devuelven el resultado de una operacion con ellos

Autor

Miguel Arconada Manteca y José Manuel Chacón Aguilera

Fecha

8-3-2018

4.10.2. Documentación de los 'defines'

4.10.2.1. BUFFER_SIZE

```
#define BUFFER_SIZE 512
```

Tamanio maximo de los buffers

4.10.2.2. ESCRITURA

```
#define ESCRITURA 1
```

Macro que define la zona de escritura de la pipe

4.10.2.3. LECTURA

```
#define LECTURA 0
```

Macro que define la zona de lectura de la pipe

4.10.2.4. MENSAJE_SIZE

```
#define MENSAJE_SIZE 512
```

Tamanio maximo de los mensajes

4.10.3. Documentación de las funciones

4.10.3.1. factorial()

```
int factorial (
            int n )
```

calcula el factorial de un numero

factorial calcula el factorial de un numero pasado como argumento. Se calcula de forma recursiva, devolviendo el numero multiplicado por el factorial del anterior, y con la condicion de parada de que factorial(1) = 1

Parámetros

arg	estructura del tipo Param que contiene la matriz, el escalar, la dimension y un identificador del hilo

Devuelve

void

4.10.3.2. main()

```
int main (
              int argc,
              char ** argv )
```

Función principal del programa.

Este programa crea cuatro procesos hijos que se comunican con el padre, e intercambian mensajes con numeros y el resultado de una operacion entre ellos diferente para cada hijo

Parámetros

argc	y argv

Devuelve

0 si todo se ejecuta correctamente, y -1 en cualquier otro caso

Índice alfabético