Sistemas Operativos, Grupo 2201 Pareja 10: Práctica 1

Generado por Doxygen 1.8.8

Viernes, 3 de Marzo de 2017 10:44:00

Índice general

Capítulo 1

Indice de archivos

1.1. Lista de archivos

Lista de todos los archivos con descripciones breves:

Capítulo 2

Documentación de archivos

2.1. Referencia del Archivo ejercicio4a.c

Sistemas Operativos: Practica 1, ejercicio 4a.

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

Dependencia gráfica adjunta para ejercicio4a.c:

'defines'

- #define NUM_PROC 3
- #define MAX_CHAR 128

Funciones

- int main (void)

  funcion main del ejercicio4a

2.1.1. Descripción detallada

Sistemas Operativos: Practica 1, ejercicio 4a.

Grupo 2201, Pareja 10. En este modulo se ha implementado el codigo del cuarto (a) ejercicio de la primera practica, referente al uso de del fork() y a los procesos huerfanos.

Autor

Blanca Martín (blanca.martins@estudiante.uam.es)
Fernando Villar (fernando.villarg@estudiante.uam.es)

Fecha

01-03-2017

2.1.2. Documentación de los 'defines'

2.1.2.1. #define MAX_CHAR 128

Maximo de caracteres para strings

2.1.2.2. #define NUM_PROC 3

Numero de iteraciones del bucle que genera procesos

2.1.3. Documentación de las funciones

2.1.3.1. int main (void)

funcion main del ejercicio4a

El programa va generando procesos en un bucle (por cada proceso existente crea un hijo) y, en esta versión, no se realiza ningún wait. Ademas, a traves del codigo contenido al final del ejercicio mostramos el arbol del proceso raiz (el padre del primer proceso) para observar si alguno de los subprocesos ha quedado huerfano.

Devuelve

EXIT_SUCCESS si se han realizado correctamente todas las tareas, EXIT_FAILURE si se ha producido algun error al ejecutar la funcion fork() o la funcion execvp.

2.2. Referencia del Archivo ejercicio4b.c

Sistemas Operativos: Practica 1, ejercicio 4b.

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

Dependencia gráfica adjunta para ejercicio4b.c:

'defines'

- #define NUM_PROC 3
- #define MAX_CHAR 128

Funciones

- int main (void)

  funcion main del ejercicio4a

2.2.1. Descripción detallada

Sistemas Operativos: Practica 1, ejercicio 4b.

Grupo 2201, Pareja 10. En este modulo se ha implementado el codigo del cuarto (a) ejercicio de la primera practica, referente al uso de del fork(), wait() y a los procesos huerfanos.

Autor

Blanca Martín (blanca.martins@estudiante.uam.es)
Fernando Villar (fernando.villarg@estudiante.uam.es)

Fecha

01-03-2017

2.2.2. Documentación de los 'defines'

2.2.2.1. #define MAX_CHAR 128

Maximo de caracteres para strings

2.2.2.2. #define NUM_PROC 3

Numero de iteraciones del bucle que genera procesos

2.2.3. Documentación de las funciones

2.2.3.1. int main (void)

funcion main del ejercicio4a

El programa va generando procesos en un bucle (por cada proceso existente crea un hijo) y, en esta versión, se realiza un solo wait por cada proceso al final. Ademas, a traves del codigo contenido al final del ejercicio mostramos el arbol del proceso raiz (el padre del primer proceso) para observar si alguno de los subprocesos ha quedado huerfano.

Devuelve

EXIT_SUCCESS si se han realizado correctamente todas las tareas, EXIT_FAILURE si se ha producido algun error al ejecutar la funcion fork() o la funcion execvp.

2.3. Referencia del Archivo ejercicio5a.c

Sistemas Operativos: Practica 1, ejercicio 5a.

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

Dependencia gráfica adjunta para ejercicio5a.c:

'defines'

- #define NUM_PROC 3

Funciones

- int main (void)

  funcion main del ejercicio5a

2.3.1. Descripción detallada

Sistemas Operativos: Practica 1, ejercicio 5a.

Grupo 2201, Pareja 10. En este modulo se ha implementado el codigo del quinto ejercicio (a) de la primera practica, referente al uso de la funcion "fork" para la creacion de conjuntos de procesos de forma secuencial o paralela.

Autor

Blanca Martín (blanca.martins@estudiante.uam.es)
Fernando Villar (fernando.villarg@estudiante.uam.es)

Fecha

01-03-2017

2.3.2. Documentación de los 'defines'

2.3.2.1. #define NUM_PROC 3

Numero de hijos a crear

2.3.3. Documentación de las funciones

2.3.3.1. int main (void)

funcion main del ejercicio5a

Este programa genera un conjunto de procesos (NUM_PROC de hijos) secuencialmente, y para cada uno imprime su ID de proceso y el ID del proceso padre.

Devuelve

EXIT_SUCCESS si se han realizado correctamente todas las tareas, EXIT_FAILURE si se ha producido algun error al ejecutar la instruccion fork().

2.4. Referencia del Archivo ejercicio5b.c

Sistemas Operativos: Practica 1, ejercicio 5b.

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

Dependencia gráfica adjunta para ejercicio5b.c:

'defines'

- #define NUM_PROC 3

Funciones

- int main (void)

  funcion main del ejercicio5b

2.4.1. Descripción detallada

Sistemas Operativos: Practica 1, ejercicio 5b.

Grupo 2201, Pareja 10. En este modulo se ha implementado el codigo del quinto ejercicio (b) de la primera practica, referente al uso de la funcion "fork" para la creacion de conjuntos de procesos de forma secuencial o paralela.

Autor

Blanca Martín (blanca.martins@estudiante.uam.es)
Fernando Villar (fernando.villarg@estudiante.uam.es)

Fecha

01-03-2017

2.4.2. Documentación de los 'defines'

2.4.2.1. #define NUM_PROC 3

Numero de hijos a crear

2.4.3. Documentación de las funciones

2.4.3.1. int main (void)

funcion main del ejercicio5b

Este programa genera un conjunto de procesos (NUM_PROC de hijos) de forma paralela, y para cada uno imprime su ID de proceso y el ID del proceso padre.

Devuelve

EXIT_SUCCESS si se han realizado correctamente todas las tareas, EXIT_FAILURE si se ha producido algun error al ejecutar la instruccion fork().

2.5. Referencia del Archivo ejercicio6.c

Sistemas Operativos: Practica 1, ejercicio 6.

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

Dependencia gráfica adjunta para ejercicio6.c:

'defines'

- #define STRING_L 80

Funciones

- int main (void)

  funcion main del ejercicio6

2.5.1. Descripción detallada

Sistemas Operativos: Practica 1, ejercicio 6.

Grupo 2201, Pareja 10. En este modulo se ha implementado el codigo del sexto ejercicio de la primera practica, referente al estudio de las consecuencias e implicaciones de la creacion de procesos hijos, en este caso en lo relacionado a recursos.

Autor

Blanca Martín (blanca.martins@estudiante.uam.es)
Fernando Villar (fernando.villarg@estudiante.uam.es)

Fecha

01-03-2017

2.5.2. Documentación de los 'defines'

2.5.2.1. #define STRING_L 80

Longitud maxima de la cadena de caracteres

2.5.3. Documentación de las funciones

2.5.3.1. int main (void)

funcion main del ejercicio6

Este programa reserva memoria para una cadena de caracteres y efectua un fork() a continuacion. EL proceso hijo escribe un valor en la cadena pasado por pantalla.

Devuelve

EXIT_SUCCESS si se han realizado correctamente todas las tareas, EXIT_FAILURE si se ha producido algun error al ejecutar la instruccion fork() o en la reserva de memoria.

2.6. Referencia del Archivo ejercicio8.c

Sistemas Operativos: Practica 1, ejercicio 8.

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

Dependencia gráfica adjunta para ejercicio8.c:

'defines'

- #define MAX_CHAR 64
- #define LENGTH_INS 2

Funciones

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

  funcion main del ejercicio8

2.6.1. Descripción detallada

Sistemas Operativos: Practica 1, ejercicio 8.

Grupo 2201, Pareja 10. En este modulo se ha implementado el codigo del octavo ejercicio de la primera practica, referente al uso de las funciones "exe" para ejecutar comandos en terminal a traves de procesos hijo.

Autor

Blanca Martín (blanca.martins@estudiante.uam.es)
Fernando Villar (fernando.villarg@estudiante.uam.es)

Fecha

01-03-2017

2.6.2. Documentación de los 'defines'

2.6.2.1. #define LENGTH_INS 2

1 (null) + longitud en palabras de instrucciones para terminal

2.6.2.2. #define MAX_CHAR 64

Maximo de caracteres para strings

2.6.3. Documentación de las funciones

2.6.3.1. int main (int *argc,* char ** *argv*)

funcion main del ejercicio8

El programa espera comandos singulares como argumentos, hace comprobacion de los mismos, y dependiendo del argumento final -l, -lp, -v o -vp, genera un proceso hijo con la funcion exe correspondiente.

Parámetros

argc,numero	de argumentos
argv,array	de strings con los argumentos

Devuelve

EXIT_SUCCESS si se han realizado correctamente todas las tareas, EXIT_FAILURE si se ha producido algun error al reservar memoria, al introducir los argumentos de entrada o al ejecutar la funcion fork().

2.7. Referencia del Archivo ejercicio9.c

Sistemas Operativos: Practica 1, ejercicio 9.

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

Dependencia gráfica adjunta para ejercicio9.c:

'defines'

- #define NUM_PROC 4
- #define MAX_CHAR 150

Funciones

- int main (void)

  funcion main del ejercicio9

2.7.1. Descripción detallada

Sistemas Operativos: Practica 1, ejercicio 9.

Grupo 2201, Pareja 10. En este modulo se ha implementado el codigo del noveno ejercicio de la primera practica, referente al uso de la funcion "pipe" para la creacion de tuberias a modo de comunicacion entre procesos padres e hijos.

Autor

Blanca Martín (blanca.martins@estudiante.uam.es)
Fernando Villar (fernando.villarg@estudiante.uam.es)

Fecha

01-03-2017

2.7.2. Documentación de los 'defines'

2.7.2.1. #define MAX_CHAR 150

Maxima longitud de cadenas de caracteres

2.7.2.2. #define NUM_PROC 4

Numero de procesos hijos a crear

2.7.3. Documentación de las funciones

2.7.3.1. int main (void)

funcion main del ejercicio9

Este programa genera un conjunto de procesos (NUM_PROC) hijos que realizan cuatro operaciones distintas↩ : suma, resta, multiplicacion y division, acorde con su orden de creacion. La comunicion entre padre e hijos se realiza a traves de una tuberia por medio de la cual el padre pasa dos operandos al proceso hijo e imprime el resultado de la operacion pertinente, devuelta por el hijo a traves de la tuberia.

Devuelve

EXIT_SUCCESS si se han realizado correctamente todas las tareas, EXIT_FAILURE si se ha producido algun error al ejecutar las instrucciones fork() o pipe().