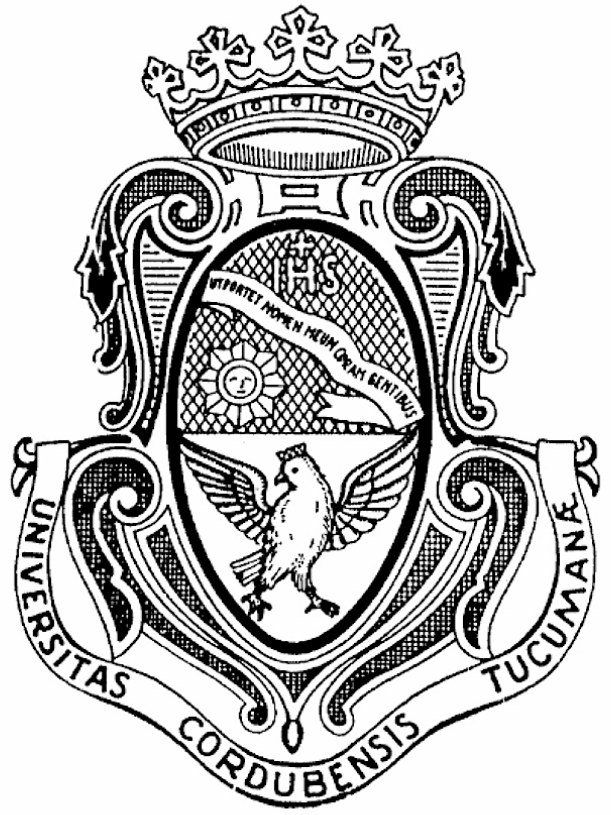
SISTEMAS OPERATIVOS I
- *2013* -

Trabajo de laboratorio 2

Baash: Una implementación del Bourne-Shell de Linux.



Alumno:
Morales, Esteban Andrés _ Mat: 35104714

Ejercicio 1:

a) Para ejecutar un archivo cuyo nombre coincide con un builtin se debe introducir precedido de la ruta de ejecución del sistema de archivos.

b) Por que si alguien copia en el directorio raíz un archivo con permisos de ejecución
que sobreescribe alguno de los programas ya existentes, Linux lo ejecutará sin importar lo que el programa realice.

c) El interprete Bash ejecuta el archivo que tiene permiso de ejecución y cuya ruta pertenece a el conjunto de rutas de PATH.

d) No existe una restricción, Bash ejecuta el comando sin ningún inconveniente.

e) Para la implementación de Baash se diseñó una rutina de ejecución que se encarga de discriminar entre comandos que ejecutan funciones builtin y programas instalados en el sistema operativo. Para estos últimos existe otra función capaz de comprobar que ruta de ejecución es válida en las contenidas en PATH para este comando.

f) Es posible modificando la variable de entorno pero esto sería poco útil ya que el propio Baash se encarga de identificar un camino relativo.

Ejercicios 2:

a) Los comandos internos para manejo de procesos de ejecución en segundo plano son: (&) y (bg luego de un Ctrl-Z).

b) Para ejecutar una secuencia de comandos en el Bourne-Shell se usa (;) entre comandos.

c) Debería imprimir 8 letras 'a'.

Ejercicios 3:

a) Tanto el operador de redireccionamiento de entrada (<) como el de salida (>) cambian las corrientes de redireccionamiento de STDOUT a un archivo y de un archivo a STDIN.

b) El redireccionamiento de entrada fue planteado de manera que el nombre de archivo se concatena a la orden de ejecución.

Ejercicios 4:

Descripción funcional de la implementación de Baash

```
char* obtener_ruta();
```

Esta función devuelve una cadena de caracteres que corresponde al PATH del sistema.

```
char** obtener_arreglo_de_rutas();
```

Esta función devuelve un arreglo de cadenas que corresponde a las rutas contempladas en el PATH del sistema.

```
int tamanio(char**);
```

Esta función calcula el tamaño en cantidad de cadenas en un arreglo de cadena de caracteres.

```
char* concatenar(char*,char*);
```

Esta función devuelve una cadena de caracteres como resultado de la concatenación de dos cadenas de caracteres en el orden de los parámetros pasados.

```
char* obtener_ruta_comando(char**, char**);
```

Esta función retorna una cadena de caracteres que corresponde a la ruta de el archivo ejecutable en el sistema de archivos.

```
char* leer_linea_terminal();
```

Esta funcion se ocupa de grabar en una cadena de caracteres lo que el usuario ha introducido por la terminal.

```
void imprimir_arreglo_de_cadenas(char**);
```

Esta es una función de control y solo se ocupa de imprimir las cadenas de caracteres de un arreglo de cadenas.

```
char** obtener_arreglo_de_argumentos(const char*);
```

Esta función devuelve un arreglo de cadenas de caracteres que corresponden a los comandos introducidos en la terminal.

```
void rutina_ejecucion();
```

Esta función se encarga de organizar la ejecución repetitiva de los servicios de Baash. Además es capaz de discernir entre tipos de ejecución, además detecta y

ejecuta funcionalidades builtin.

```
void cambiar_directorio(char**);
```

Implementación de la funcionalidad de comando 'cd'.

```
void ejecutar_detras(char**);
```

Esta función se encarga de ejecutar comandos en segundo plano siempre y cuando se haya detectado la orden con el comando '&' entre los comandos.

```
void ejecutar_estandar(char**);
```

Esta función se encarga de ejecutar todos los comandos introducidos por terminal y que no corresponden a funcionalidades builtin. Además detecta el comando de ejecución en segundo plano y deriva su operatividad. Finalmente es capaz de ejecutar lineas de comando con redirección en ambos sentidos.

```
char* cadena_final(char**);
```

Esta función devuelve la última cadena de caracteres de un arreglo de cadenas.

```
char caracter_final(char*);
```

Análogo a la función anterior retorna el último caracter de una cadena.

```
char** modificar_argumentosxejecucion_detras(char**);
```

Esta función devuelve el arreglo de comandos a ejecutarse en segundo plano, esto es sin el operador '&' sea cual fuere la posición del mismo en el arreglo de comandos original.

```
char** tipo_redireccion(char**);
```

Esta función devuelve un par de cadenas de caracteres que corresponden a el tipo de redirección con un número del 1 al 5 y el operador.

```
char** modificar_argumentosxredireccion(char**,
char**,char**);
```

Esta función devuelve el arreglo de comandos a ejecutarse en el caso de existir una solicitud de redirección '<' ó '>' en el arreglo de comandos original.

```
char* detectar_redireccion(char**);
```

Esta función devuelve una cadena con un único caracter identificativo de redirección, esto es "<" ó bien ">".

```
int contar_caracter_enArreglo_deCadenas(char, char**);
```

Esta función cuenta la cantidad de ocurrencias de cierto caracter en un arreglo de cadenas.

```
int tamanio_enCaracteres(char**);
```

Esta función retorna el tamaño de un arreglo de cadenas en cantidad de caracteres total.

```
char** copia_arreglo_deCadenas(char**);
```

Esta función hace una copia en la memoria de un arreglo de cadenas de caracteres.

```
int error_en_redireccion(char** argumentos);
```

Esta función verifica si la solicitud de redirección fue ingresada debidamente para ello usa una bandera de tipo entero que retorna para su posterior control.

```
par_ejecucion obtener_comandos_ejecucionxtuberias(char*);
```

Esta función devuelve una estructura definida como el tipo par_ejecución que consta de dos arreglos de cadenas que corresponden a el comando y los argumentos que ejecutará el proceso padre como así también el comando y los argumentos que ejecutará el proceso hijo respectivamente en caso de solicitud de redirección mediante 'pipes'.

```
void ejecutar_tuberias(par_ejecucion);
```

Esta función se encarga de crear los procesos necesarios para una ejecución con redirección por 'pipes'.

```
void imprimir_spot();
```

Esta función imprime el encabezado de ejecución y presentación de Baash.

```
void imprimir_ayuda();
```

Esta función imprime la ayuda.