SISTEMAS OPERATIVOS

Grado en Informática. Curso 2015-2016

Práctica 1

`PARTE A`

Realizar un intérprete de comandos sencillo (shell) en UNIX. El intérprete debe comprender los comandos que se muestran a continuación.

fin Termina la ejecución del intérprete de comandos.

exit Termina la ejecución del intérprete de comandos.

quit Termina la ejecución del intérprete de comandos.

autores Muestra los nombres y los logins de los autores del shell.

pid [-p] Muestra el pid del proceso que ejecuta el shell (con -p muestra también el del proceso padre del shell)

cd [dir] Cambia el directorio actual del shell a *dir*. Si no se especifica *dir*, muestra el directorio actual del shell.

list [-s] [-a] [dir] Lista los ficheros del directorio *dir*, excluyendo los archivos ocultos (aquéllos cuyo nombre comienza por '.'). Para cada fichero nos dará una línea con la información en el formato análogo a como lo hace `ls -li`. Si no se especificase *dir* se listará el directorio actual. La opción *-s* indica listado corto, es decir, para cada fichero se listará solo su nombre. La opción *-a* indica que se listen también los archivos cuyo nombre comienza por '**.**'

delete archivo Elimina *archivo*. *archivo* es un fichero o un directorio vacio.

deltree dir Elimina el directorio *dir* junto con todos los ficheros y directorios que contenga. *dir* es un directorio y puede o no estár vacio

`PARTE B`

- Utlizando las funciones hechas en el apartado anterior realizar dos programas independientes, *list* y *deltree*, que hagan lo mismo que los correspondientes comandos del shell, pero recibiendo sus argumentos de línea de comando

Además, deben tenerse en cuenta las siguientes indicaciones:

- **En ningún caso debe producir un error de ejecución (segmentation, bus error ...). La práctica que produzca un error en tiempo de ejecución no será puntuada.**

- No debe dilapidar memoria (ejemplo: variable que se asigna cada vez que se llama a una función y no se libera). **NO SE REFIERE A DECLARAR LOS ARRAYS DE TAMAÑO PEQUEÑO** (puede utilizarse *valgrind* para detectar errores de memoria)
- Cuando el shell no pueda ejecutar una acción por algún motivo, debe indicarlo con un mensaje como el que se obtiene con `sys_errlist[errno]` o con *perror()* (por ejemplo, si no puede cambiar de directorio debe indicar por qué).

AYUDAS

Un shell es basicamente un bucle que realiza lo siguiente

```
....
while (!terminado){
   imprimirPrompt();
   leerEntrada();
   procesarEntrada();
}
.....
```

imprimirPrompt() y *leerEntrada()* pueden ser algo tan sencillo como sendas llamadas a `printf` y `gets`

El primer paso para procesar la entrada es trocear la cadena de entrada. Para trocear la cadena de entrada es muy cómodo usar `strtok`. Téngase en cuenta que `strtok` ni asigna memoria ni copia cadenas, solo pone caracteres de fin de cadena en determinados sitios de la cadena de entrada. La siguiente función trocea la cadena de entrada (se supone que no recibe un puntero NULL) en un array de punteros terminado a NULL (el último puntero del array es NULL). Devuelve el número de trozos

```
int TrocearCadena(char * cadena, char * trozos[])
{
  int i=1;

  if ((trozos[0]=strtok(cadena," \n\t"))==NULL)
    return 0;
  while ((trozos[i]=strtok(NULL," \n\t"))!=NULL)
    i++;
  return i;
}
```

Para convertir los permisos de un fichero a la cadena que los representa, puede

utilizarse cualquiera de estas funciones, téngase en cuenta que *ConvierteModo2* es la versión con asignación estática de memoria y *ConvierteModo3* es la versión con asignación dinámica de memoria

```
char TipoFichero (mode_t m)
{
  switch (m&S_IFMT) { /*and bit a bit con los bits de formato,0170000 */
    case S_IFSOCK:  return 's';  /*socket */
    case S_IFLNK:   return 'l';    /*symbolic link*/
    case S_IFREG:   return '-';   /* fichero normal*/
    case S_IFBLK:   return 'b';   /*block device*/
    case S_IFDIR:   return 'd';   /*directorio */
    case S_IFCHR:   return 'c';   /*char  device*/
    case S_IFIFO:   return 'p';   /*pipe*/
    default: return '?';   /*desconocido, no deberia aparecer*/
  }
}

char * ConvierteModo (mode_t m, char *permisos)
{
  strcpy (permisos,"---------- ");

  permisos[0]=TipoFichero(m);
  if (m&S_IRUSR) permisos[1]='r';  /*propietario*/
  if (m&S_IWUSR) permisos[2]='w';
  if (m&S_IXUSR) permisos[3]='x';
  if (m&S_IRGRP) permisos[4]='r';   /*grupo*/
  if (m&S_IWGRP) permisos[5]='w';
  if (m&S_IXGRP) permisos[6]='x';
  if (m&S_IROTH) permisos[7]='r';   /*resto*/
  if (m&S_IWOTH) permisos[8]='w';
  if (m&S_IXOTH) permisos[9]='x';
  if (m&S_ISUID) permisos[3]='s';  /*setuid, setgid y stickybit*/
  if (m&S_ISGID) permisos[6]='s';
  if (m&S_ISVTX) permisos[9]='t';
  return permisos;
}

char * ConvierteModo2 (mode_t m)
{
  static char permisos[12];
  strcpy (permisos,"---------- ");
```

```
  permisos[0]=TipoFichero(m);
  if (m&S_IRUSR) permisos[1]='r';  /*propietario*/
  if (m&S_IWUSR) permisos[2]='w';
  if (m&S_IXUSR) permisos[3]='x';
  if (m&S_IRGRP) permisos[4]='r';   /*grupo*/
  if (m&S_IWGRP) permisos[5]='w';
  if (m&S_IXGRP) permisos[6]='x';
  if (m&S_IROTH) permisos[7]='r';   /*resto*/
  if (m&S_IWOTH) permisos[8]='w';
  if (m&S_IXOTH) permisos[9]='x';
  if (m&S_ISUID) permisos[3]='s';  /*setuid, setgid y stickybit*/
  if (m&S_ISGID) permisos[6]='s';
  if (m&S_ISVTX) permisos[9]='t';
  return (permisos);
}

char * ConvierteModo3 (mode_t m)
{
  char * permisos;
  permisos=(char *) malloc (12);
  strcpy (permisos,"---------- ");

  permisos[0]=TipoFichero(m);
  if (m&S_IRUSR) permisos[1]='r';  /*propietario*/
  if (m&S_IWUSR) permisos[2]='w';
  if (m&S_IXUSR) permisos[3]='x';
  if (m&S_IRGRP) permisos[4]='r';   /*grupo*/
  if (m&S_IWGRP) permisos[5]='w';
  if (m&S_IXGRP) permisos[6]='x';
  if (m&S_IROTH) permisos[7]='r';   /*resto*/
  if (m&S_IWOTH) permisos[8]='w';
  if (m&S_IXOTH) permisos[9]='x';
  if (m&S_ISUID) permisos[3]='s';  /*setuid, setgid y stickybit*/
  if (m&S_ISGID) permisos[6]='s';
  if (m&S_ISVTX) permisos[9]='t';
  return (permisos);
}
```

IMPORTANTE: FORMA DE ENTREGA

Las práctica se realizará en **GRUPOS DE DOS ALUMNOS** y se en-

tregará mediante el repositorio de *subversion* bajo el directorio P1 antes de proceder a su defensa. Esta carpeta deberá incluir tanto el código fuente como el fichero Makefile que permite su compilación, si lo hubiere. Las cabeceras de los ficheros fuente deben incluir un comentario con los nombres de los integrantes del grupo de prácticas y el horario que les corresponde.

Uno de los integrantes del grupo será el responsable del grupo y **SOLO ÉL ENTREGARÁ LAS PRÁCTICAS**. El responsable del grupo lo será para todas las prácticas. Práctica entregada por más de un componente del grupo será clasificada por los scripts de comprobación como copia y por tanto conllevará la pérdida de la calificación en prácticas.

La práctica será defendida ante el profesor en el aula y en horario de prácticas. Todos los miembros del grupo deberán estar presentes para la defensa, de forma que el profesor pueda revisar su funcionamiento así como realizar comentarios/cuestiones a los integrantes del grupo o pedir cambios en el código que se puedan considerar pertinentes.

Fecha límite de subida al *svn*: (23h, 59m del Viernes 9 Octubre. DEFENSA DE LA PRACTICA semana 12-16/Octubre/2015