CC3301 Programación de Software de Sistemas

Control 2 – Semestre Primavera 2012

Prof.: Luis Mateu

Pregunta 1

Parte a.- Programe la función:

```
void invocar_fun( char *name,
                  void (*fun)(char *name,
                              struct stat *s) );
```

El primer parámetro corresponde a un nombre que puede ser un directorio o un archivo regular. El segundo parámetro es un puntero a la función *fun*. Si se trata de un archivo regular, *invocar_fun* debe llamar a *fun* pasándole como argumentos el nombre del archivo y sus características recopiladas con la función *stat*. Si se trata de un directorio, se debe llamar recursivamente *invocar_fun* para cada uno de los elementos de ese directorio. *Invocar_fun* debe ignorar los archivos que no son directorios o no son archivos regulares, como por ejemplo los links simbólicos.

Parte b.- Escriba un programa que use la función *invocar_fun* para determinar el archivo regular más grande que se ubique en algún lugar dentro de la jerarquía de archivos y directorios accesibles desde el directorio actual. El programa debe mostrar el nombre de ese archivo y su tamaño. Ejemplo de uso:

```
% ./mas_grande ~jgonzalez
video.mpg 1250456293
%
```

Pregunta 2

Parte a.- Escriba un programa que recibe desde la línea de comandos un conjunto de archivos con la extensión ".c". Su programa debe usar fork para compilar en paralelo todos estos archivos usando "`gcc -c` *nombre-de-archivo*". Es decir Ud. *no debe* esperar a que un gcc termine para iniciar un nuevo gcc. Ejemplo de uso:

```
% gcc_paralelo hello.c jalisco.c gato.c
%
```

Antes de terminar, su programa debe asegurarse de que todas las compilaciones hayan finalizado. Cuando un archivo tiene errores de compilación, el comando gcc envía los mensajes de error a la salida estándar de errores (descriptor de archivo número 2). Su programa también debe enviar los mensajes de error a la salida estándar de errores, sin preocuparse de que aparezcan mezclados los mensajes de error de un archivo con los mensajes de error de otros archivos.

Parte b.- Modifique su programa haciendo ahora que los mensajes de error *no aparezcan mezclados*. La salida de errores debe tener el siguiente formato:

```
% gcc_paralelo hello.c jalisco.c gato.c
=== hello.c ===
hello.c:3 hello undeclared
...
=== jalisco.c ===
jalisco.c:5 unterminated string jalisco nunca pierde
...
=== gato.c ===
...
%
```

Ayuda de memoria:

```
char *path= "/usr/local";
struct stat sbuf;
if (stat(path, &sbuf)==0) ...
if (S_ISDIR(sbuf.st_mode)) ...        si path es un directorio
if (S_ISREG(sbuf.st_mode)) ...        si path es un archivo regular
ssize_t s= sbuf.st_size;              tamaño del archivo

if (chdir(path)==0) ...

DIR *dir= opendir(path);
struct dirent *ent= readdir(dir);
char *name= ent->d_name;                nombre del archivo
closedir(dir);

pid_t pid= fork();
waitpid(pid, NULL, 0);
execlp("ls", "ls", "-l", NULL);

int fd= open("aux.txt", "r");
char buf[80];
ssize_t n= read(fd, buf, 80);
close(fd);

int fd_a[2];
if (pipe(fd_a)==0) ...
int nuevo_fd= dup(fd_a[1]);
FILE *file= fdopen(nuevo_fd, "w");     para usar fprintf
fprintf(file, "%d", s);
fclose(file);
```