¿Cómo se usa feof() y ferror() con fgets()?

Las funciones feof() y ferror(), comprueban los indicadores de final de fichero y error, respectivamente, de un stream de entrada. Dichos indicadores se activan o desactivan después de cada operación con el correspondiente stream. Por lo tanto, feof() no comprueba si se alcanzará el EOF del stream, sino si se alcanzó en la última operación con el mismo. De forma parecida, ferror() comprueba si se encontró un error en la última operación con el stream.

De este modo, un bucle de lectura debe comprobar si se ha alcanzado el fin del fichero o si se ha producido un error depués de la operación sobre un stream, no antes. Si se hace de otra manera puede intentarse procesar una cadena de entrada que no se ha podido obtener, por ejemplo. Considerensé los siguientes ejemplos, el de la izquierda está mal, los otros dos bien:

MAL	BIEN	MEJOR

/* Esta solución no es buena,
porque no se asegura de que no
se haya encontrado un EOF antes
de procesar la nueva línea leída*/
while (!feof (fch))
{
fgets(lna, MAX_LNA, fch);
/*puede que se encuentre un EOF al leer, entonces en lna estará lo mismo que antes de leer*/
sscanf(lna,"%d|%d",&i,&j);
}
	/* Esta solución resuelve los problemas de la mala, pero no es eficiente porque hace dos comprobaciones por bucle*/
	while (!feof (fch))
	{
	if (fgets(lna, MAX_LNA, fch))
	{
	sscanf(lna,"%d|%d",&i,&j);
	}
	}
		while (fgets(lna, MAX_LNA, fch))
		/*cuando hay error o se encuentra EOF fgets devuelve 0*/
		{ /*siempre que se entra aquí hay
		un valor nuevo en lna*/
		sscanf(lna,"%d|%d",&i,&j);
		}
		if(!feof(fch)) /*si se ha terminado el
		bucle y feof devuelve 0, se ha producido un error*/
		{
		printf("Error: Lectura fichero");
		} /*si no, es que el bucle se ha terminado, porque se ha llegado al EOF*/

TIPO DE DATO	DESCRIPCIÓN	TAMAÑO	OBSERVACIONES
char	un caracter alfanumérico	1 Byte	Aunque representan caracteres alfanuméricos, un char es un entero sin signo de un byte de tamaño. Estos caracteres corresponden a los de la tabla ASCIIl
int	un numero entero	4 Bytes *	
float	punto flotante (con decimales)	4 Bytes *	útil para números grandes, pero poca precisión, no adecuado para muchos decimales
double	punto flotante con doble precisión	8 Bytes *	más precisión, pero ocupa más memoria

MODIFICADORES	DESCRIPCIÓN	TAMAÑO	OBSERVACIONES
short	Modifican el tamaño del int por defecto	16 bits *	Deben cumplir que short int <= int <= long int <= long long int.
long		32 bits *	
signed	Tipos de datos con signo (positivo y negativo)		Si no se indica nada, por defecto se considera que el int es signed int.
unsigned	Tipos de datos sin signo (solo positivo)		

Comentarios	Aunque se tratan de modificadores, pueden considerarse ellos solos como tipo. Si no se aplican a ningún tipo de dato se considera por defecto que se aplica al tipo int, es decir, que las siguientes sentencias son equivalentes.
	
	`int a;`
	`short int a;`
	`short a;`
	`signed a;`
	`signed int a;`
	
	Y lo mismo ocurre con long y unsigned.