

INSTITUTO POLITECNICO NACIONAL.

ESCUELA SUPERIOR DE CÓMPUTO.

Título: Investigación.

Alumno: Angel Miranda Pedro.

Materia: Embedded systems.

Grupo: 3CV10.

Profesor: García Ortega Víctor Hugo.

**El uso de caracteres Wilcard**

Wilcard (comodín)

Algunas veces, se necesita ejecutar un comando sobre varios archivos al mismo tiempo. Un claro ejemplo de esto es el comando ls, el cual lista información acerca de los ficheros alojados en algún directorio especificado.

Los nombres de archivo son tan importantes en ambientes Linux, que las shells proveen un mecanismo para poder especificar patrones de nombres de archivos sin tener que conocer el nombre de estos en su totalidad. Se pueden utilizar caracteres especiales llamados comodines (wildcards en inglés) utilizados para representar patrones de caracteres.

Un solo nombre de archivo puede especificar muchos archivos usando caracteres comodines. Los caracteres comodín en make son '\*', '?' Y '[...]', al igual que en el shell Bourne. Por ejemplo, \* .c especifica una lista de todos los archivos (en el directorio de trabajo) cuyos nombres terminan en '.c'.

El carácter '~' al comienzo de un nombre de archivo también tiene un significado especial. Si está solo o seguido de una barra inclinada, representa el directorio de inicio. Por ejemplo, ~ / bin se expande a / home / you / bin. Si al '~' le sigue una palabra, la cadena representa el directorio de inicio del usuario nombrado por esa palabra. Por ejemplo ~ john / bin se expande a / home / john / bin. En sistemas que no tienen un directorio de inicio para cada usuario (como MS-DOS o MS-Windows), esta funcionalidad se puede simular estableciendo la variable de entorno HOME.

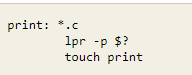
La expansión de comodines se realiza automáticamente en objetivos y en requisitos previos. En las recetas, el shell es responsable de la expansión del comodín. En otros contextos, la expansión de comodines ocurre solo si la solicita explícitamente con la función comodín.

El significado especial de un carácter comodín se puede desactivar precediéndolo con una barra diagonal inversa. Por lo tanto, foo \ \* bar se referiría a un archivo específico cuyo nombre consiste en 'foo', un asterisco y 'bar'.

Los comodines se pueden usar en la receta de una regla, donde el caparazón los expande. Por ejemplo, aquí hay una regla para eliminar todos los archivos de objeto:



Los comodines también son útiles en los requisitos previos de una regla. Con la siguiente regla en el archivo MAKE, 'make print' imprimirá todos los archivos '.c' que han cambiado desde la última vez que los imprimió:



La expansión del comodín no ocurre cuando defines una variable. Por lo tanto, si escribes esto



El valor de los objetos variables es la cadena real '\* .o'. Sin embargo, si usa el valor de los objetos en un objetivo o prerrequisito, la expansión del comodín tendrá lugar allí. Si usa el valor de los objetos en una receta, el intérprete de comandos puede realizar la expansión de comodines cuando se ejecuta la receta. Para establecer objetos para la expansión, en su lugar use:



**Funciones para análisis y sustitución de cadenas.**

Aquí hay algunas funciones que operan en cadenas

$(subst from,to,text)

Realiza un reemplazo de texto en el texto de texto: cada aparición de from se reemplaza por to. El resultado se sustituye por la llamada a la función. Por ejemplo.



sustituye la cadena 'fEEt on the strEEt'.

$(patsubst pattern,replacement,text)

Encuentra palabras separadas por espacios en blanco en texto que coincide con el patrón y las reemplaza con reemplazo. Aquí el patrón puede contener un '%' que actúa como un comodín, haciendo coincidir cualquier número de caracteres dentro de una palabra. Si el reemplazo también contiene un '%', el '%' se reemplaza por el texto que coincide con el '%' en el patrón. Solo el primer '%' en el patrón y reemplazo se trata de esta manera; cualquier '%' posterior no cambia.

Los caracteres '%' en las invocaciones de la función patsubst se pueden citar con las barras diagonales anteriores ('\'). Las barras diagonales inversas que de lo contrario citarían caracteres '%' pueden citarse con más barras invertidas. Las barras invertidas que citan '%' caracteres u otras barras invertidas se eliminan del patrón antes de que se comparen los nombres de los archivos o se le sustituye por un vástago. Las barras invertidas que no están en peligro de citar caracteres '%' no se ven afectadas. Por ejemplo, el patrón \%% raro \\% patrón \\ tiene '%% raro' antes del carácter operativo '%' y 'patrón \\' lo sigue. Las dos últimas barras invertidas se dejan en blanco porque no pueden afectar ningún carácter '%'.

El espacio en blanco entre palabras se pliega en caracteres de espacio único; los espacios en blanco iniciales y finales se descartan.

Por ejemplo.



Produce el valor 'x.c.o bar.o'.

Las referencias de sustitución (ver Referencias de sustitución) son una forma más sencilla de obtener el efecto de la función patsubst:



Es equivalente a



La segunda taquigrafía simplifica uno de los usos más comunes de patsubst: reemplazar el sufijo al final de los nombres de los archivos.



Es equivalente a



Por ejemplo, puede tener una lista de archivos de objetos:



Para obtener la lista de archivos fuente correspondiente, simplemente puede escribir:



en lugar de usar la forma general:

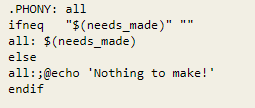


$(strip string)

Elimina los espacios en blanco iniciales y finales de la cadena y reemplaza cada secuencia interna de uno o más caracteres en espacios en blanco con un solo espacio. Por lo tanto, '$ (strip a b c)' da como resultado 'a b c'.

La tira de función puede ser muy útil cuando se usa junto con condicionales. Al comparar algo con la cadena vacía '' usando ifeq o ifneq, generalmente desea una cadena de espacios en blanco para que coincida con la cadena vacía (consulte Condición).

Por lo tanto, los siguientes pueden no tener los resultados deseados:



Reemplazar la referencia de variable '$ (needs\_made)' con la llamada de función '$ (strip $ (needs\_made))' en la directiva ifneq lo haría más robusto.

Busca una ocurrencia de hallazgo. Si ocurre, el valor es encontrar; de lo contrario, el valor está vacío. Puede usar esta función en un condicional para probar la presencia de una subcadena específica en una cadena dada. Por lo tanto, los dos ejemplos.



Produce los valores 'a' y '' (la cadena vacía), respectivamente. Ver Banderas de prueba, para una aplicación práctica de la cadena de descubrimiento.

$(filter pattern…,text)

Devuelve todas las palabras separadas por espacios en blanco en el texto que coinciden con cualquiera de las palabras de patrón, eliminando cualquier palabra que no coincida. Los patrones se escriben usando '%', al igual que los patrones utilizados en la función de patsubst anterior.

La función de filtro se puede utilizar para separar diferentes tipos de cadenas (como nombres de archivos) en una variable. Por ejemplo:

