Examen 103 - Comandos GNU y Unix

103.2 Procesar flujos de texto usando filtros

Teoría

En el mundo Unix/Linux, muchos comandos están diseñados para funcionar como "filtros": leen datos de una entrada estándar (stdin), procesan esos datos de alguna manera y escriben el resultado en una salida estándar (stdout). Los mensajes de error suelen enviarse a una salida de error estándar (stderr). Esta arquitectura permite combinar comandos de forma muy potente (ver 103.4 - Redirecciones y Pipes).

Flujos Estándar:

- **stdin (Standard Input):** Por defecto, la entrada del teclado. Los comandos leen desde aquí.
- **stdout (Standard Output):** Por defecto, la salida a la pantalla/terminal. Los resultados de los comandos se envían aquí.
- **stderr (Standard Error):** Por defecto, también la salida a la pantalla/terminal. Los mensajes de error se envían aquí. Se mantiene separado de stdout para poder manejar errores de forma distinta a la salida normal.

Filtros Comunes y su Propósito:

Aquí tienes una descripción de algunos comandos muy comunes que actúan como filtros:

1. **`cat` (concatenate):**
    - Muestra el contenido de archivos. Lee archivos y los escribe a stdout.
    - Si se usa sin argumentos de archivo, lee de stdin y lo escribe a stdout (útil para introducir texto desde el teclado hasta que pulses Ctrl+D, fin de entrada).
    - `cat archivo1 archivo2`: Muestra el contenido de ambos archivos secuencialmente.
    - `cat -n archivo`: Muestra el contenido con números de línea.
2. **`less`:**
    - Un paginador de texto. Permite ver archivos largos pantalla a pantalla.
    - Lee de stdin (si se usa en una pipe, ej: `ls -l | less`) o de archivos especificados como argumentos.
    - Permite buscar texto, navegar hacia arriba/abajo, etc. (Usa las flechas, `Page Up`/`Page Down`, `/` para buscar, `q` para salir). Es una mejora sobre el antiguo comando `more`.
3. **`head`:**
    - Muestra las primeras líneas de un archivo o de stdin.
    - `head archivo`: Muestra las primeras 10 líneas por defecto.
    - `head -n 20 archivo`: Muestra las primeras 20 líneas.
    - `head -c 100 archivo`: Muestra los primeros 100 bytes/caracteres.
4. **`tail`:**

- Muestra las últimas líneas de un archivo o de stdin.
- `tail archivo`: Muestra las últimas 10 líneas por defecto.
- `tail -n 20 archivo`: Muestra las últimas 20 líneas.
- `tail -f archivo`: Muestra las últimas líneas y luego "sigue" el archivo, mostrando nuevas líneas a medida que se añaden (útil para monitorizar archivos de log).

5. **`sort`:**
   - Ordena las líneas de un archivo o de stdin.
   - `sort archivo`: Ordena alfabéticamente (por defecto) y escribe a stdout.
   - `sort -r archivo`: Ordena en orden inverso.
   - `sort -n archivo`: Ordena numéricamente.
   - `sort -k 2 archivo`: Ordena basándose en la segunda columna/campo.
   - `sort -u archivo`: Ordena y elimina líneas duplicadas (como `sort | uniq`).
6. **`uniq`:**
   - Reporta o elimina líneas duplicadas **adyacentes** de un archivo o stdin.
   - `uniq archivo`: Elimina líneas adyacentes duplicadas. **Importante:** Si quieres eliminar *todas* las duplicadas, primero debes ordenar el archivo (`sort archivo | uniq`).
   - `uniq -c archivo`: Cuenta las ocurrencias de cada línea única adyacente.
   - `uniq -d archivo`: Muestra solo las líneas duplicadas adyacentes.
   - `uniq -u archivo`: Muestra solo las líneas únicas (no duplicadas adyacentes).
7. **`wc` (word count):**
   - Cuenta líneas, palabras y caracteres de un archivo o stdin.
   - `wc archivo`: Muestra líneas, palabras y bytes.
   - `wc -l archivo`: Cuenta solo líneas.
   - `wc -w archivo`: Cuenta solo palabras.
   - `wc -c archivo`: Cuenta solo bytes.
8. **`grep` (Global Regular Expression Print):**
   - Busca líneas en un archivo o stdin que coincidan con un patrón (una expresión regular - ver 103.7).
   - `grep 'patron' archivo`: Muestra las líneas que contienen 'patron'.
   - `grep -v 'patron' archivo`: Muestra las líneas que *no* contienen 'patron'.
   - `grep -i 'patron' archivo`: Busca sin distinguir mayúsculas/minúsculas.
   - `grep -n 'patron' archivo`: Muestra las líneas encontradas con su número de línea.
   - `grep -r 'patron' directorio`: Busca recursivamente en archivos dentro de un directorio.
9. **`sed` (Stream EDitor):**
   - Un editor de flujos no interactivo. Puede realizar transformaciones de texto básicas (sustitución, eliminación de líneas, etc.) basadas en patrones.

- `sed 's/patron/reemplazo/g' archivo`: Reemplaza todas las ocurrencias de 'patron' por 'reemplazo' en cada línea.

10. **awk:**
    - Un lenguaje de programación diseñado para procesar texto basado en columnas o campos. Permite realizar tareas más complejas como formatear salida, calcular sumas, etc.
    - `awk '{ print $1 }' archivo`: Imprime la primera columna de cada línea.
    - `awk '$3 > 100 { print $1 }' archivo`: Imprime la primera columna de las líneas donde la tercera columna es mayor que 100.

Estos comandos, al leer de stdin y escribir a stdout, son perfectos para ser encadenados usando pipes (`|`) o para ser utilizados con redirecciones (`>` o `<`), lo que veremos en el objetivo 103.4.