Trabajo Final Integrador 2022

Seminario de Lenguajes - Opción “C” 2022

Desarrollar una versión reducida de los programas Unix `head` y `tail`.

Las fechas de entrega, coloquio de defensa y re-entrega se publicarán por la plataforma de Cátedras.

Para la aprobación del trabajo, el mismo debe cumplir con lo solicitado en el enunciado, estar correctamente modularizado y no presentar errores (de lógica, de acceso a memoria, de acceso a archivos, etc). Notar que hay puntos del enunciado que deben realizarse únicamente en el caso de haber desaprobado algún parcial durante la cursada.

El trabajo consiste en la implementación en lenguaje C de dos programas disponibles en los sistemas Unix y derivados: `head` y `tail`. El funcionamiento principal de estos comandos es, respectivamente, mostrar las primeras o últimas *N* líneas de un archivo de texto. Mediante el uso de argumentos se puede personalizar este funcionamiento, e inclusive utilizarlos sobre la entrada estándar del programa en lugar de realizar la operación sobre archivos regulares del sistema de archivos.

A continuación se describen los argumentos que cada uno de estos programas a desarrollar deben admitir, junto con el resultado que tienen sobre el funcionamiento del mismo.

Comando `head`

Funcionamiento

Este programa deberá imprimir en la salida estándar las primeras *N* líneas del archivo a procesar.

Sinopsis

```
head [-l LINEAS | -b BYTES] [ARCHIVO]
```

Argumentos

Se admitirán los siguientes argumentos:

- `-l LINEAS` ó `--lines LINEAS`
  Indica la cantidad de líneas a imprimir. Si se omite y tampoco se especifica el argumento `-b`, su valor por defecto será definido en tiempo de compilación (llamaremos a este valor `LINEAS_POR_DEFECTO`, y en caso de no especificarse en tiempo de compilación su valor por defecto será 5). Es mutuamente excluyente con el argumento `-b`.
  Si el archivo a procesar tuviese menos líneas que el valor de `LINEAS`, se imprimirán tantas líneas como el archivo contenga.

- `-b BYTES` ó `--bytes BYTES`
  Indica la cantidad de bytes, contados desde el comienzo del archivo, que se deben imprimir. Si se omite, se asume que el archivo se procesará por líneas. Es mutuamente excluyente con el argumento `-l`.
  Si el archivo a procesar tuviese menos bytes que el valor de `BYTES`, se imprimirán tantos bytes como el archivo contenga.
- `ARCHIVO`
  Ruta al archivo a procesar. Si se omite, se utilizará la entrada estándar del programa.

El programa deberá terminar con un valor de retorno adecuado al resultado de su ejecución, considerando distintos valores de retorno para distintas situaciones de error. Los mismos deberán estar documentados.

Ejemplos

```
head
```

Imprime las primeras `LINEAS_POR_DEFECTO` líneas que se reciban por la entrada estándar.

```
head -l 25
```

Imprime las primeras 25 líneas que se reciban por la entrada estándar.

```
head -b 100
```

Imprime los primeros 100 bytes que se reciban por la entrada estándar.

```
head /var/log/nginx/error.log
```

Imprime las primeras `LINEAS_POR_DEFECTO` líneas del archivo `/var/log/nginx/error.log`.

```
head --lines 120 archivo.txt
```

Imprime las primeras 120 líneas del archivo `archivo.txt`.

```
head --bytes 50 directorio/archivo.txt
```

Imprime los primeros 50 bytes del archivo `directorio/archivo.txt`.

Ítem adicional para quien haya desaprobado al menos una evaluación

Se deberá extender el funcionamiento del comando para admitir múltiples archivos de entrada, siendo la siguiente la sinopsis resultante para el programa:

```
head [-l LINEAS | -b BYTES] [ARCHIVO_1 ARCHIVO_2 … ARCHIVO_N]
```

De esta forma, el programa admitirá que se indique más de un archivo a procesar, en cuyo caso la salida del mismo deberá modificarse para incluir un encabezado con el nombre de cada archivo antes de su contenido.
Por ejemplo, la siguiente invocación del programa:

```
head -l 1 introduccion.txt capitulo_1.txt capitulo_2.txt
```

Producirá la siguiente salida:

```
*** introduccion.txt ***
Esta es la primera linea del archivo introduccion.txt.

*** capitulo_1.txt ***
Esta es la primera linea del archivo capitulo_1.txt.

*** capitulo_2.txt ***
Esta es la primera linea del archivo capitulo_2.txt.
```

Nota: el encabezado con el nombre de cada archivo se debe incluir únicamente cuando se vaya a procesar más de un archivo de entrada.

Comando `tail`

Funcionamiento

Este programa deberá mostrar las últimas *N* líneas del archivo a procesar.

Sinopsis

```
tail [-l LINEAS] [-r] [ARCHIVO]
```

Argumentos

Se admitirán los siguientes argumentos:

- `-l LINEAS` ó `--lines LINEAS`
  Indica la cantidad de líneas a imprimir. Si se omite, su valor por defecto será definido en tiempo de compilación (llamaremos a este valor `LINEAS_POR_DEFECTO`, y en caso de no especificarse en tiempo de compilación su valor por defecto será 5).
  Si el archivo a procesar tuviese menos líneas que el valor de `LINEAS`, se imprimirán tantas líneas como el archivo contenga.
- `-r` ó `--reverse`
  Indica que las líneas deben imprimirse en orden inverso al que poseen en el

archivo. Esto es: cada línea se imprimirá tal cual se lee, pero comenzando por la última línea del archivo e imprimiendo luego las anteriores.

- `ARCHIVO`
  Ruta al archivo a procesar. Si se omite, se utilizará la entrada estándar del programa.

El programa deberá terminar con un valor de retorno adecuado al resultado de su ejecución, considerando distintos valores de retorno para distintas situaciones de error. Los mismos deberán estar documentados.

Ejemplos

```
tail
```

Imprime las últimas `LINEAS_POR_DEFECTO` líneas que se reciban por la entrada estándar.

```
tail -l 25
```

Imprime las últimas 25 líneas que se reciban por la entrada estándar.

```
tail -r
```

Imprime las últimas `LINEAS_POR_DEFECTO` que se reciban por la entrada estándar en orden inverso.

```
tail /var/log/nginx/access.log
```

Imprime las últimas `LINEAS_POR_DEFECTO` líneas del archivo `/var/log/nginx/access.log`.

```
tail --lines 120 archivo.txt
```

Imprime las últimas 120 líneas del archivo `archivo.txt`.

```
tail -l 5 --reverse directorio/archivo.txt
```

Imprime las últimas 5 líneas del archivo `directorio/archivo.txt` en orden inverso.

Ítem adicional para quien haya desaprobado al menos una evaluación

Se deberá extender el funcionamiento del comando para admitir múltiples archivos de entrada, siendo la siguiente la sinopsis resultante para el programa:

```
tail [-l LINEAS] [-r] [ARCHIVO_1 ARCHIVO_2 … ARCHIVO_N]
```

De esta forma, el programa admitirá que se indique más de un archivo a procesar, en cuyo caso la salida del mismo deberá modificarse para incluir un encabezado con el nombre de cada archivo antes de su contenido.
Por ejemplo, la siguiente invocación del programa:

```
tail -l 1 introduccion.txt capitulo_1.txt capitulo_2.txt
```

Producirá la siguiente salida:

```
*** introduccion.txt ***
Esta es la última linea del archivo introduccion.txt.

*** capitulo_1.txt ***
Esta es la última linea del archivo capitulo_1.txt.

*** capitulo_2.txt ***
Esta es la última linea del archivo capitulo_2.txt.
```

Nota: el encabezado con el nombre de cada archivo se debe incluir únicamente cuando se vaya a procesar más de un archivo de entrada.

Modalidad de entrega

Se deberán entregar todos los archivos fuente que hayas generado para desarrollar los programas, utilizando correcta modularización, y un archivo README que contenga una breve descripción del proyecto, decisiones importantes de diseño que hayas tomado, e instrucciones para compilar ambos comandos.