CC3301 Programación de Software de Sistemas – Semestre Otoño 2025
Tarea 2 – Profs. Mateu/Ibarra/Urrea

En esta tarea Ud. deberá programar 2 funciones para invertir todas las palabras de un string, eliminando los espacios redundantes. Las 2 funciones pedidas son:

```
void invertir(char *str);
char *invertido(char *str);
```

La función *invertir* altera el mismo string *str* que recibe como parámetro; mientras que la función *invertido* retorna un nuevo string resultante sin alterar el parámetro *str* original.

Ambas funciones deben invertir la posición de cada palabra que se encuentre dentro del string *str*, dejándolas separadas únicamente por un espacio. Los siguientes son ejemplos de uso de la función *invertir* para distintos *str*.

Llamada	***str inicial***	***Resultado en str***
`invertir(str);`	`"hola que tal"`	`"tal que hola"`
`invertir(str);`	`"  hola que tal"`	`"tal que hola"`
`invertir(str);`	`"hola que tal  "`	`"tal que hola"`
`Invertir(str);`	`" hola  que  tal "`	`"tal que hola"`

Restricción:

- No use el operador de subindicación de arreglos *[ ]* ni su equivalente **(p+i)*, en su lugar use aritmética de punteros, privilegiando los operadores ++ -- += -=. Sí puede usar *p+i* o *p-i*.
- En *invertir* no use *malloc* para pedir memoria auxiliar. Para poder invertir el orden de cada palabra dentro del string se le recomienda primero invertir el string original completamente, por ejemplo sea *str = "a ab cd"* entonces obtenga el string *"dc ba a".* Una vez realizado dicho paso, vaya palabra por palabra del string invertido y vuelva a invertirlas una a una de forma independiente, por ejemplo sea el string invertido *"dc ba a"* entonces tome cada palabra por si sola y vuelva a invertirlas en el mismo lugar, es decir *"dc"* se pasa a *"cd", "ba"* se pasa a *"ab",* quedando el string final como *"cd ab a"*.
- En *invertido* sí debe usar *malloc* para pedir el espacio justo ocupado por el resultado final.
- Es importante que tanto para *invertir* como *invertido* se preocupe de eliminar los espacios redundantes en algún momento del procedimiento.
- Tenga cuidado de no pasarse de los límites del string original.

Instrucciones

Baje *t2.zip* de U-cursos y descomprímalo. El directorio *T2* contiene entre otros archivos (a) *test-invertir.c* que prueba si su tarea funciona y compara su eficiencia con la solución del profesor, (b) *prof.ref-x86_64 y prof.ref-aarch64* con los binarios ejecutables de la solución del profesor, (c) *invertir.h* que incluye el encabezado de la función pedida, y (d) *Makefile* que le servirá para compilar y ejecutar su tarea. Ud. debe programar las funciones solicitadas en el archivo *invertir.c*.

Pruebe su tarea bajo Debian 12 nativo o virtualizado con VirtualBox, Vmware, QEmu o WSL 2. Ejecute el comando *make* sin parámetros. Le mostrará las opciones que tiene para compilar su tarea. Estos son los requerimientos para aprobar su tarea:

- ***make run* debe felicitarlo por aprobar este modo de ejecución. Su solución no debe ser 80% más lenta que la solución del profesor.**
- ***make run-g* debe felicitarlo.**
- *make run-san* debe felicitarlo y no reportar ningún problema como por ejemplo goteras de memoria.

Cuando pruebe su tarea con *make run* asegúrese que su computador esté configurado en modo alto rendimiento y que no estén corriendo otros procesos intensivos en uso de CPU al mismo tiempo. De otro modo podría no lograr la eficiencia solicitada.

Entrega

Ud. solo debe entregar por medio de U-cursos el archivo *invertir.zip* generado por el comando *make zip*. **A continuación es muy importante que descargue de U-cursos el mismo archivo que subió, luego descargue nuevamente los archivos adjuntos y vuelva a probar la tarea tal cual como la entregó**. Esto es para evitar que Ud. reciba un 1.0 en su tarea porque entregó los archivos equivocados. Créame, sucede a menudo por ahorrarse esta verificación. Se descontará medio punto por día de atraso. No se consideran los días de receso, sábados, domingos o festivos.