

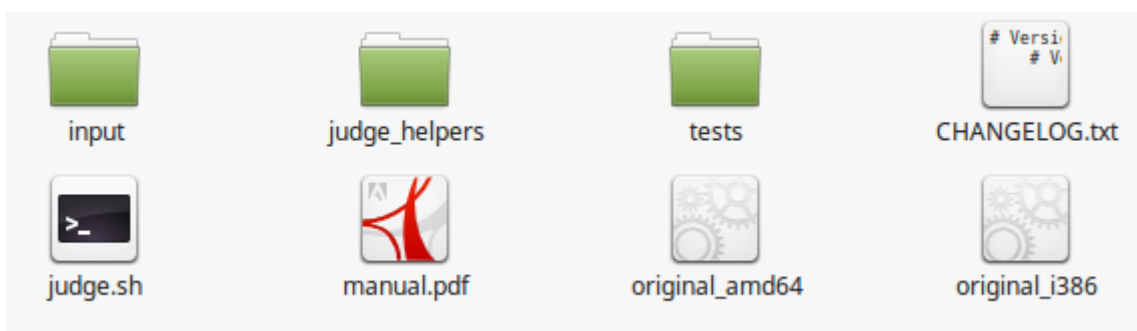
INFO

Este programa es un judge al igual que [JAIME](#), o [SPOJ](#) y [CODEFORCES](#) para los que alguna vez hicieron problemas tipo [ACM-ICPC](#). Esto significa que el Judge funciona tomando dos programas de entradas, uno que se considera “correcto” y otro que se intenta evaluar; y compara la salidas que producen ambos ante un mismo input, por lo que si bien no es 100% perfecto y no reemplaza la ejecución a mano, facilita y acelera mucho la detección de errores en el programa a evaluar.

Este Judge es un script [BASH](#), por lo que debería ser compatible con todas las diferentes distribuciones de GNU con Linux. Al estar escrito en este lenguaje, no es necesario compilarlo para poder ejecutarlo y el código puede ser modificado con cualquier editor de textos.

EJECUCIÓN

Una vez descomprimido el archivo .zip, deberían obtener una carpeta con los siguientes archivos:



input: La carpeta que contiene los diccionarios entregados por los profesores de la materia

judge_helpers: Archivos extra necesarios para la ejecución del Judge

tests: La carpeta que contiene los test cases a probar por el Judge

CHANGELOG.txt: Archivo que contiene el registro de cambios entre las versiones del Judge

judge.sh: El script principal del Judge

manual.pdf: Este archivo

original_amd64: El programa compilado a partir de los código objeto entregados por los profesores para procesadores de 64bits (Nueva versión)

original_i386: El programa compilado a partir de los código objeto entregados por los profesores para procesadores de 32bits (Nueva versión)

Para ejecutar el Judge por primera vez, lo más probable es que primero necesiten darle permisos de ejecución, desde la terminal esto se puede hacer navegando hasta la carpeta donde se encuentra el Judge y ejecutando:

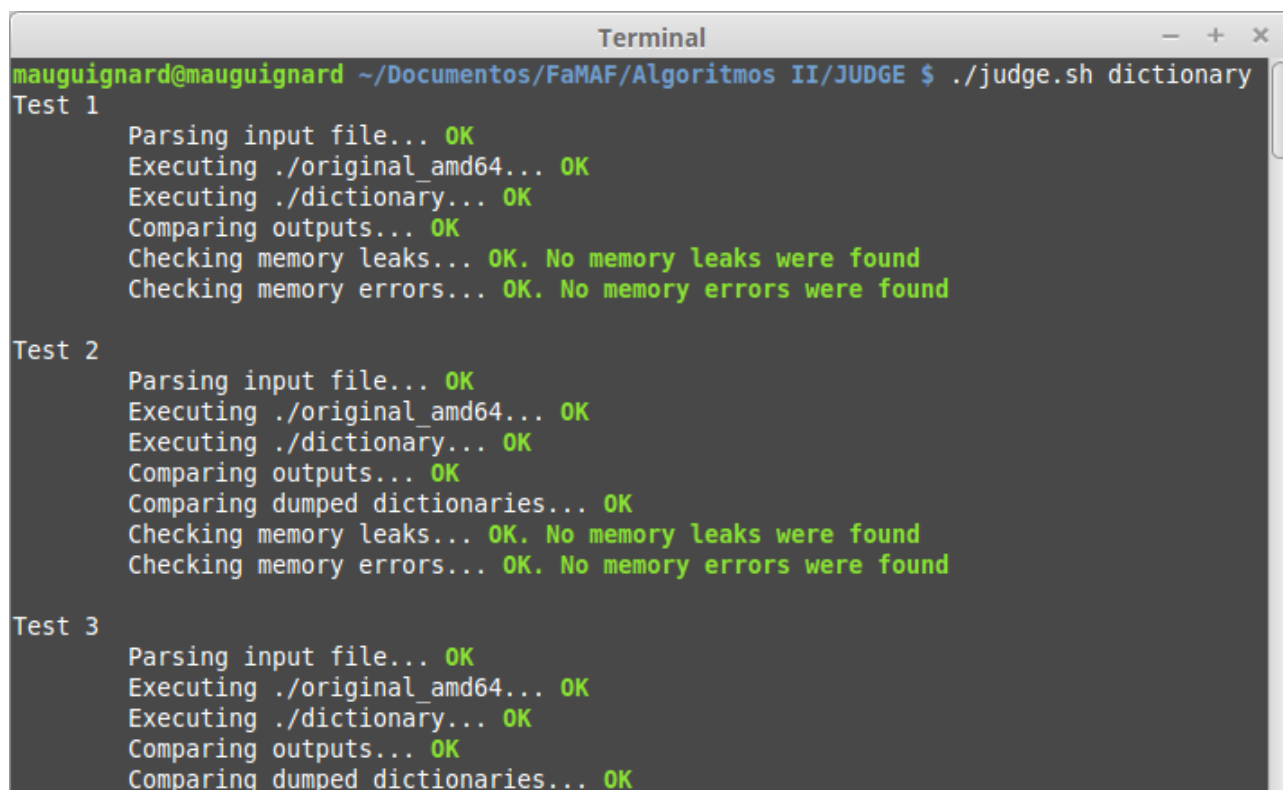
```
chmod +x judge.sh
```

Cuando deseen correr el Judge para comprobar si su programa es correcto y no tiene errores y/o memory leaks, compilen y luego copien su aplicación a la carpeta donde se encuentra el Judge y desde la terminal ejecútenlo usando este comando:

```
./judge.sh dictionary
```

Donde “dictionary” es la aplicación compilada a partir del código escrito por ustedes.

Si todo sale bien, el Judge debería ejecutarse y mostrarles los resultados:



```
Terminal
mauguignard@mauguignard ~/Documentos/FaMAF/Algoritmos II/JUDGE $ ./judge.sh dictionary
Test 1
  Parsing input file... OK
  Executing ./original_amd64... OK
  Executing ./dictionary... OK
  Comparing outputs... OK
  Checking memory leaks... OK. No memory leaks were found
  Checking memory errors... OK. No memory errors were found

Test 2
  Parsing input file... OK
  Executing ./original_amd64... OK
  Executing ./dictionary... OK
  Comparing outputs... OK
  Comparing dumped dictionaries... OK
  Checking memory leaks... OK. No memory leaks were found
  Checking memory errors... OK. No memory errors were found

Test 3
  Parsing input file... OK
  Executing ./original_amd64... OK
  Executing ./dictionary... OK
  Comparing outputs... OK
  Comparing dumped dictionaries... OK
```

IMPORANTE: El Judge para decidir si la salida de la aplicación es correcta, se limita a comparar textualmente la salida del programa de los profes con el de ustedes. Por lo tanto, si su main genera un menú diferente al de los profes, por más que el programa de ustedes ande correctamente, el Judge va a decir WRONG ANSWER.

IMPORTANTE: El Judge sólo puede comparar las salidas que generan los diferentes test cases que se encuentran en la carpeta tests, por lo tanto mientras más test cases tengan y más complejos sean éstos, más probabilidades hay de que el Judge detecte algún error en caso de que exista. El Judge incluye 17 test cases propios que tratan de probar la mayor cantidad de casos que se puedan presentar, pero ustedes pueden modificarlos o incluso agregar los suyos propios. Además, también incluye 4 test cases adicionales que son los fixtures que usa [JAIME](#) para los test de aceptación (main).

IMPORTANTE: El Judge NO es un reemplazo de [JAIME](#). La idea es que corran el Judge en sus computadoras y una vez que éste diga que su programa funciona correctamente y no tiene memory leaks, recién ahí lo suban a [JAIME](#), ya que éste sólo les dejará probar su código unas 10 veces.

POSIBLES RESPUESTAS

Parsing input file...

OK = El archivo del test case fue procesado correctamente.

ERROR = Hubo algún problema procesando el test case.

Executing ./original_amd64...

Executing ./original_i386...

Executing ./dictionary...

OK = El programa se ejecutó correctamente.

TIME LIMIT EXCEEDED = El programa superó el tiempo máximo determinado para su ejecución (Predeterminado: 30 segundos). Si sólo su programa dio TIME LIMIT EXCEEDED (TLE), lo más probable es que haya entrado en un bucle infinito. Sin embargo, también puede deberse al hecho que el Judge sólo corre Valgrind en la aplicación de ustedes, el cual hace que la ejecución de la aplicación sea mucho más lenta al tener que analizar toda la memoria que utiliza la aplicación en busca de errores o memory leaks. Si sólo obtienen TLE en uno o dos test cases, pueden probar aumentando el tiempo máximo permitido de ejecución modificando la variable MAX_TIME en el archivo judge.sh.

Por el contrario, si obtienen TLE también en la aplicación de los profes, lo más probable es que alguno de test cases (los archivos en la carpeta tests) esté fallado.

ERROR = Hubo un error al ejecutar la aplicación.

Comparing outputs...

Comparing dumped dictionaries...

OK = Las salidas de ambos programas son idénticas.

PRESENTATION ERROR = Las salidas de ambos programas son idénticas, excepto capitalización y/o espaciado.

Ejemplo:

Si la salida del programa de los profes es "The size is 0." y la salida del programa de ustedes es " the size is 0 ."

WRONG ANSWER = Las salidas de los programas son diferentes entre sí.

Checking memory leaks...

OK = No hubo memory leaks durante la ejecución de su programa.

WRONG ANSWER = Hubo 1 o más memory leaks.

Checking memory errors...

OK = No hubo ningún intento de escritura/lectura de memoria inválido.

WRONG ANSWER = Hubo 1 o más accesos (lectura/escritura) inválidos a la memoria.

DETECTANDO ERRORES

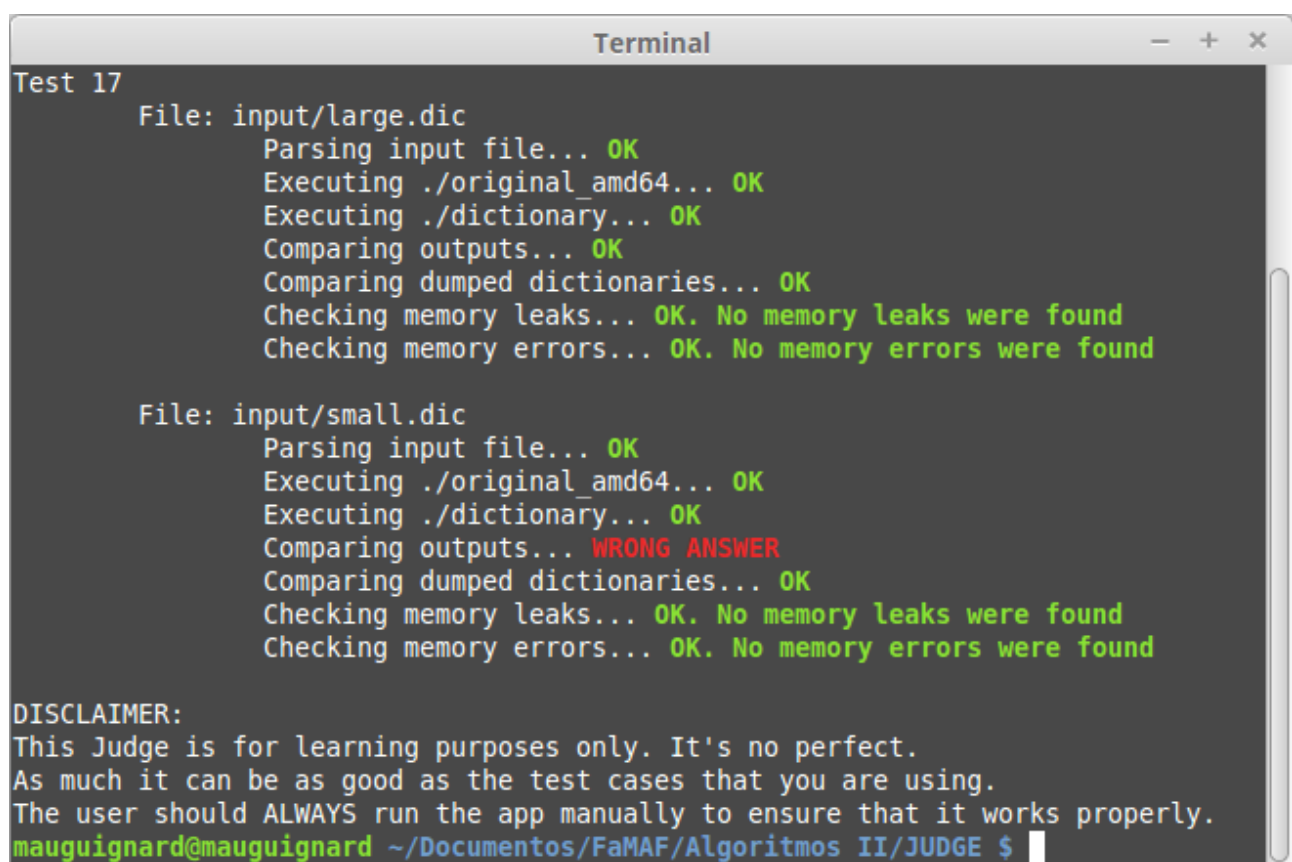
Ya que el Judge no muestra que diferencias existen entre las salidas de los 2 programas, éste soporta ejecutar sólo un test case en particular para luego comparar las salidas a mano.

En particular, tiene dos tipos de filtros:

- t | --test Sólo ejecuta el test case cuyo número sea igual al especificado
- f | --file En los test cases que dependan de los diccionarios de la carpeta input, el Judge sólo los va a ejecutar para el archivo especificado

Estos filtros pueden combinarse entre sí.

Por ejemplo, si al ejecutar el Judge obtienen la siguiente salida:



```
Terminal
Test 17
  File: input/large.dic
    Parsing input file... OK
    Executing ./original_amd64... OK
    Executing ./dictionary... OK
    Comparing outputs... OK
    Comparing dumped dictionaries... OK
    Checking memory leaks... OK. No memory leaks were found
    Checking memory errors... OK. No memory errors were found

  File: input/small.dic
    Parsing input file... OK
    Executing ./original_amd64... OK
    Executing ./dictionary... OK
    Comparing outputs... WRONG ANSWER
    Comparing dumped dictionaries... OK
    Checking memory leaks... OK. No memory leaks were found
    Checking memory errors... OK. No memory errors were found

DISCLAIMER:
This Judge is for learning purposes only. It's no perfect.
As much it can be as good as the test cases that you are using.
The user should ALWAYS run the app manually to ensure that it works properly.
mauguignard@mauguignard ~/Documentos/FaMAF/Algoritmos II/JUDGE $
```

Pueden ejecutar el Judge de nuevo pero sólo probando el Test case 17 para el archivo small.dic escribiendo lo siguiente en la Terminal:

```
./judge.sh --test=17 --file=small.dic dictionary
```

Para lo cual, en este supuesto caso, el Judge debería devolver lo siguiente:

```
Terminal
mauguignard@mauguignard ~/Documentos/FaMAF/Algoritmos II/JUDGE $ ./judge.sh --test=17 --file=small.dic dictionary
Test 17
  File: input/small.dic
    Parsing input file... OK
    Executing ./original_amd64... OK
    Executing ./dictionary... OK
    Comparing outputs... WRONG ANSWER
    Comparing dumped dictionaries... OK
    Checking memory leaks... OK. No memory leaks were found
    Checking memory errors... OK. No memory errors were found

DISCLAIMER:
This Judge is for learning purposes only. It's not perfect.
As much as it can be as good as the test cases that you are using.
The user should ALWAYS run the app manually to ensure that it works properly.

Remove temporary files? [y/n] n
mauguignard@mauguignard ~/Documentos/FaMAF/Algoritmos II/JUDGE $
```

Notar que al ejecutar un sólo test case, al terminar el Judge pregunta si los archivos temporales creados deberían ser eliminados. Para poder comparar manualmente las salidas de los programas es importante que pongan no (n).

Luego, pueden comparar los archivos a mano usando diff, meld o alguna herramienta similar.

```
Terminal
mauguignard@mauguignard ~/Documentos/FaMAF/Algoritmos II/JUDGE $ diff TRUSTED_OUTPUT.txt TO_CHECK_OUTPUT.txt
281c281
< Please enter your choice:      -> The size is 2.
---
> Please enter your choice:      -> The size is 3.
mauguignard@mauguignard ~/Documentos/FaMAF/Algoritmos II/JUDGE $
```

