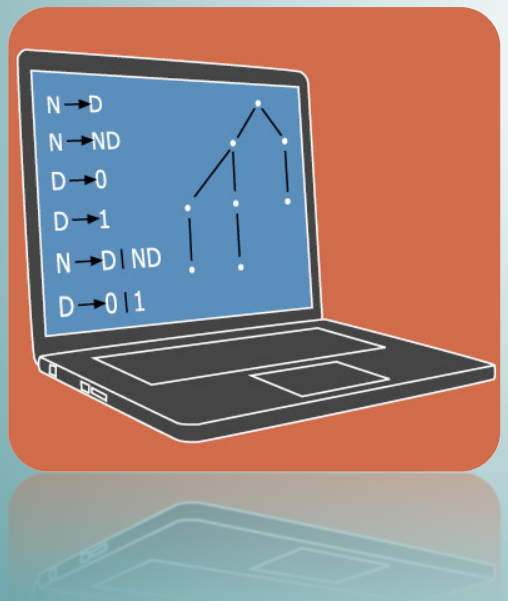
# Procesadores de Lenguaje

## Presentación




**Cristina Tirnauca**

DPTO. DE MATEMÁTICAS,

ESTADÍSTICA Y COMPUTACIÓN

# Generalidades

## El diseño de la asignatura: Prof. José L. Balcázar (UPC)

1. Cuestiones cotidianas.
2. Evaluación.
3. El nombre de la asignatura.
4. Descripción y motivación.
5. Contenidos.
6. Bibliografía.

# Cuestiones Cotidianas

## De índole práctica

### Personal e infraestructura

- Clases a cargo de: Domingo Gómez (profesor responsable), Cristina Tîrnăucă.
- Desarrollo teórico, ejemplos y ejercicios, en general los miércoles (11:45 a 13:45) y jueves (12:45 a 13:45) - aula 4.
- Sesiones de problemas (puntuables) los miércoles (Grupo 1) y los viernes (Grupo 2) de 08:30 a 09:30 (LSC-I).
- Laboratorio los martes (15:30 a 17:30, LSC-I): sesiones de prácticas (puntuables) hasta aproximadamente Semana Santa; desarrollo de una práctica individual a partir de ese punto.
- `ocw.unican.es`
  `moodle.unican.es`
  `personales.unican.es/gomezd`
  `personales.unican.es/tirnaucac`.

# Evaluación (I)

## Atención, que es complicada!

A lo largo del curso se obtiene una nota de curso en $[0, 6]$.

En el examen final se obtienen una nota básica de examen en $[0, 6]$ y una nota de profundización en $[0, 4]$, mediante preguntas diferenciadas.

### La nota de curso depende:

- de la práctica individual que realices,
- de tu participación en el grupo de trabajo, y
- de tu aportación a determinadas sesiones de problemas.

Cada sesión puntuable de problemas aporta muy poquito, pero, si trabajas un poco la asignatura, todas juntas se notarán.
Cabe la posibilidad de llegar al examen final sabiendo que tienes la asignatura aprobada.

# Evaluación (II)

## Si tu nota de curso es baja, no ocurre nada grave

### Nota final:

Suma de la nota básica más la nota de profundización.

- La nota de profundización se obtiene en el examen final, respondiendo a las preguntas que se especificarán como válidas para tal fin.
- La nota básica se obtiene truncando a 6 la suma de la nota de curso y la nota que se obtenga en el resto de las preguntas del examen final.

# ¿Procesadores de Lenguaje?

## ¿Qué será "procesar"? ¿Qué "lenguaje"?

Grupo central de conceptos y técnicas que tienen en común dos hechos concretos:

- esperar unos datos inherentemente secuenciales,
- y buscar relaciones entre ellos que pueden requerir salvar distancias no acotadas.

¡Estos conceptos se pueden aplicar en muchos ámbitos!

- Principal: la Compilación, tratamiento de Lenguajes de Programación.
- Aquí desarrollaremos más la parte llamada Compilación Frontal (*Front-End Compiler*).
- Trataremos en menor profundidad, al final del curso, cuestiones relacionadas.

# Convenciones sobre lenguajes

## Con alguna que otra duda

¿A qué llamamos un lenguaje? ¿Y una lengua? ¿Y un idioma? ¿Y un dialecto? ¿Cuándo está bien definido un lenguaje?

### Convención:

Distinción entre

- "Lenguajes naturales" y
- "Lenguajes formales",

distinción profundamente discutible: hemos de aceptarla por el contexto en que vivimos, pero ¡no nos dejemos enredar!

### Observación crucial:

- Un programa puede procesar una secuencia de datos.
- Un programa es una secuencia de datos.
- Por tanto, un programa puede procesar otros programas.
- ¡O incluso procesarse a sí mismo!

# El proceso de Compilación (I)

## Fases, pasos y *scans*

Estructura y tipologías de compiladores:

1. monopaso y multipaso;
2. multipaso y/o multi-*scan*;
3. fases:
   - análisis léxico,
   - análisis sintáctico,
   - análisis semántico,
   - optimización de código,
   - generación de código;
4. front-end (código fuente $\longrightarrow$ código intermedio) y back-end (código intermedio $\longrightarrow$ código máquina).

# El proceso de Compilación (II)

## Paso y *scan* no es lo mismo

### División de la compilación en pasos:

Cada paso trabaja sobre un objeto que ha dejado en disco el paso anterior, y deja en disco un objeto para el paso siguiente.

- El primero de esos objetos es el programa fuente.
- El último es el ejecutable (o interpretable).

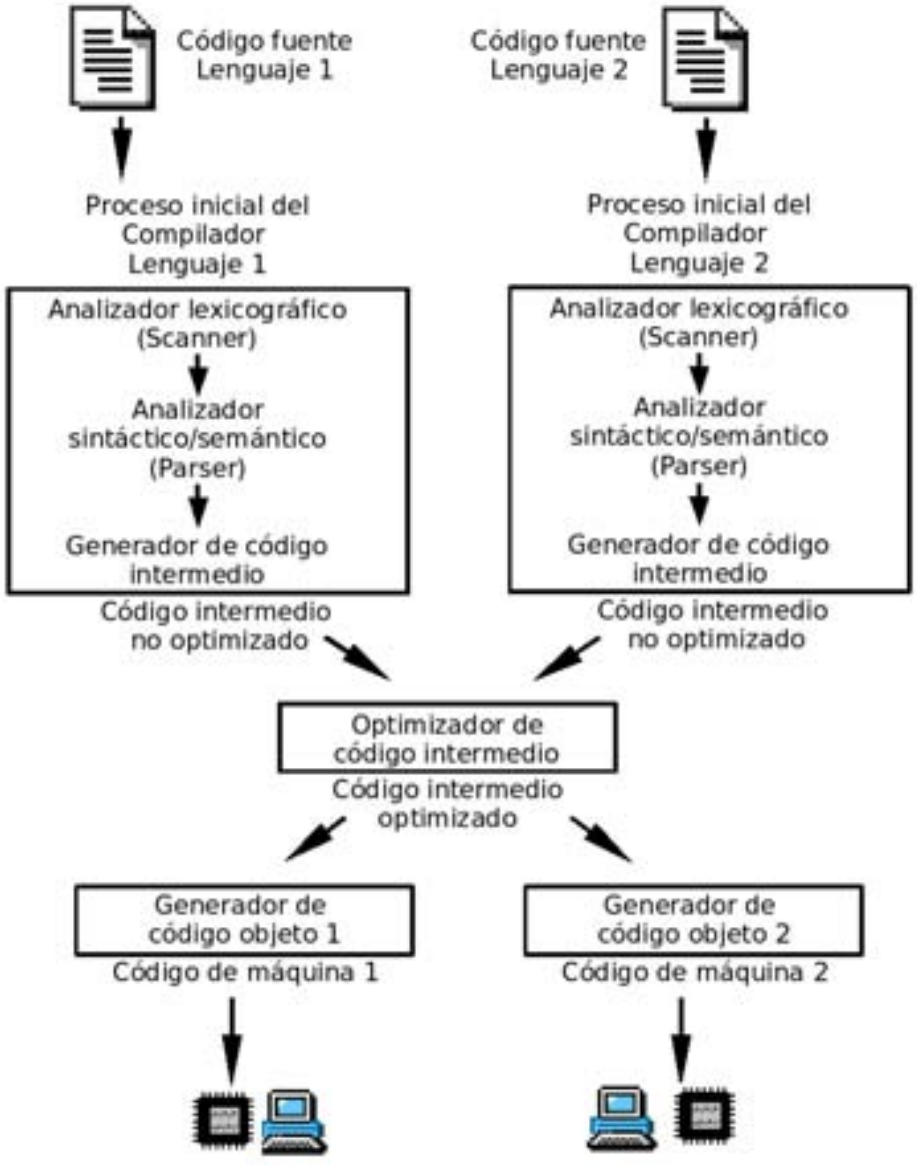
### *Scans*:

Recorridos del programa fuente.

- Hay como mínimo uno.
- Pueden convenir más de uno:
  - Declaraciones anticipadas (por ejemplo recursividad mutua),
  - Directivas de compilación...
- Puede haber un solo *scan* y luego otros pasos.

# El proceso de Compilación (III)

## Fases clásicas (Fuente: Wikipedia)



# Objetivos

## ¿Por qué aprender sobre procesadores de lenguajes?

### ¡Una de las asignaturas más instructivas!

- Esencialmente informática.
- Excepcional equilibrio entre teoría y práctica.
- Intuiciones entre las más valiosas de la informática:
  - conciencia de las distinciones entre signos, símbolos y significados,
  - confianza en el manejo de la recursividad,
  - capacidad de salvar varios niveles de abstracción de una tacada.
- Cuando sabes bien lo que hace el compilador,
  - tu eficacia como programador mejora,
  - y tu eficacia como coordinador de programadores mejora mucho.
- Cada vez más, el procesamiento de lenguajes es crucial en la informática empresarial (dialectos de XML).

# El punto de partida

## Malas noticias... ;-)

El punto de partida de esta asignatura es la Teoría de Autómatas y Lenguajes Formales.

¡Nadie dijo que dominar Compiladores fuese fácil!

(Pero no nos vamos a asustar por eso ahora, ¿no?)

Nuestro propósito:

- Convenceros de que el esfuerzo es asequible
  (si nos curramos con ganas la asignatura).
- Convenceros de que el esfuerzo merece la pena.
- Ofreceros una oportunidad de preparación en un tema fascinante para un informático, que os haga preparar la asignatura como si no hubiera un examen final que aprobar.
- Motivaros para dedicar unas cuantas horas más a programar
  (que siempre habrán sido pocas).

# Procesos básicos

## En el procesamiento de lenguajes

### Es útil referirse a varios grados de abstracción:

1. Codificaciones y manipulación carácter a carácter; ejemplo: UTF-8.
2. Identificación de símbolos: caracteres o secuencias (breves) de caracteres a las que se dota, por convención humana, de un significado; ejemplos: Sí, while, %, <=.
3. Estructura de los símbolos (frecuentemente organizados de manera secuencial) y relaciones entre ellos (análisis sintáctico); ejemplos: sintagmas nominales, concordancias.
4. Reajuste de los significados de los símbolos en función de las relaciones estructurales y generación de nuevo significado; ejemplo: resolución de ambigüedades ("María guardó las revistas que Paco dejó bajo la cama").
5. Expresión del mismo significado empleando una convención diferente; ejemplo: traducciones.

# El analizador léxico

También *tokenizer*, *lexer* o *scanner*

Recompone, a partir de los caracteres individuales, los símbolos básicos con significado bien definido.

## Su entrada:

El fuente del programa a compilar. Lo lee carácter a carácter.

## Su salida:

Las "piezas" con significado que lo conforman (*tokens*).

Conceptualmente es relativamente sencillo; tecnológicamente es uno de los puntos más delicados porque no es fácil hacerlo rápido.

Solemos emplear un programa que recibe la especificación de las "piezas" autorizadas en el lenguaje (en la forma de expresiones regulares extendidas) y genera el programa analizador. En C/C++ se trata de `lex` o `flex`.

# Lenguajes regulares

## De lo más útil para quien los sabe usar

### Equilibrio entre sencillez y potencia.

Ideales para:

1. Análisis léxico: identificación de qué secuencias de caracteres hemos de considerar agrupadas con un único significado; ejemplo: "SELECT".
2. Transformaciones muy sencillas que se requiera realizar para operar con información que resida en ficheros texto; ejemplo: ¿campos separados por comas o por tabuladores?
3. Búsquedas en textos; ejemplo: operador "/" en el comando "less".
4. Preproceso, reformateado y limpieza de datos en proyectos de Minería de Datos.

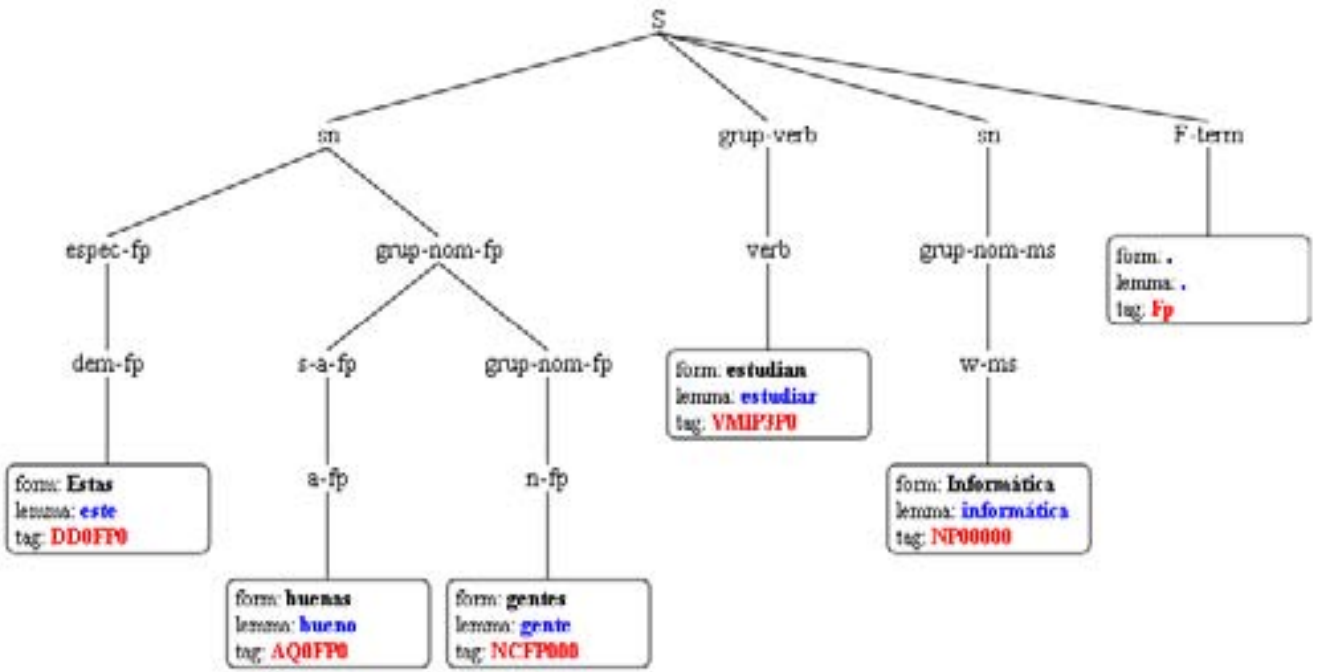
# El análizador sintáctico (I)

## La piedra angular

En lengua inglesa "to parse":

> "In linguistics, to parse is to divide language into its grammatical components that can be analyzed. For example, dividing a sentence into verbs, adjectives, nouns, and so on."

Llevamos toda la vida haciendo análisis sintáctico.




`http://www.lsi.upc.edu/ nlp/semeval/docs/tagset-constituents.es.pdf`

# El análizador sintáctico (II)

O *parser*

Identifica las relaciones estructurales entre las "piezas".

## Su entrada:

La secuencia de piezas producida por el analizador léxico.

## Su salida:

Frecuentemente, el árbol sintáctico que corresponde; a veces, el árbol no llega a materializarse sino que el analizador interactúa con el análisis semántico y genera directamente un código intermedio.

Conceptualmente es algo más complicado que el análisis léxico, porque se trata de construir un autómata de pila que funcione en tiempo lineal.

Solemos emplear un programa que recibe la especificación de las "piezas" autorizadas en el lenguaje y de la gramática que lo define, y genera el programa analizador y su interacción con el analizador léxico. En C/C++ se trata de `yacc` o `bison`.

# El análizador sintáctico (III)

"Syntax error"

## Crucial en muchas aplicaciones:

1. Por supuesto Compiladores,
2. aplicaciones que incluyen algo de lenguas humanas (NLP),
3. pero la transformación de árboles sintácticos es la clave para el buen funcionamiento de los optimizadores de SQL,
4. y la disponibilidad de analizadores y la sencillez impuesta a su sintaxis son la clave para la amplia extensión reciente de XML.

Los analizadores sintácticos ("parsers") más eficaces corresponden de manera muy precisa a los lenguajes incontextuales o de tipo 2 de la Jerarquía de Chomsky (en concreto, al subtipo determinista).

Compensamos sus limitaciones ampliando con cuidado la capacidad de los analizadores sintácticos y diseñando procesos adicionales.

# El analizador semántico

## Conjunto o no con el sintáctico

Se ocupa de todo aquello que no se puede expresar mediante un lenguaje incontextual determinista:

- verificación de tipos (type checking),
- condiciones de declaración (se asocian las referencias a variables y funciones con sus definiciones),
- generación de código.

Solemos emplear opciones adicionales del programa (`yacc` o `bison` para C/C++) que genera el analizador sintáctico para que éste implemente también el análisis semántico, o bien para que construya en memoria un árbol sintáctico abstracto que el analizador semántico recorrerá.

# Contenidos (I)

## Teóricos y prácticos

### Teóricos:

- analizadores sintácticos ascendentes (LR, SLR, LALR);
- otros analizadores (léxicos, sintácticos descendentes: predictivos y con retroceso . . . );
- gramáticas de atributos, y su uso para mantener tablas de símbolos y generar código intermedio;
- compilación cruzada y *bootstrapping* de compiladores (diagramas en T).
- perspectivas adicionales en función del desarrollo del curso (XML, serialización, código nativo, máquinas virtuales, procesado de lenguas humanas, compilación de lenguajes declarativos u orientados a objetos,. . . )

# Contenidos (II)

## Teóricos y Prácticos

### Prácticos:

Herramientas: Python, NLTK, C++, Bison, Flex, ...

Trabajo en grupo:

- grupos de 4-5 personas
- varios roles: búsqueda de información, elaboración de documentos: informe/lista de problemas, presentación oral (teoría y práctica),
- la nota se obtiene haciendo una media (autoevaluación, co-evaluación y evaluación del profesor) y es la misma para todo el grupo
- rúbrica para la evaluación
- cuestionario para formar los grupos

Práctica individual:

- Invéntate un lenguaje sencillo y construye el "compilador".

# Capacidades a Desarrollar

(Por vuestra parte, no por la nuestra)

## De varios niveles:

1. Manejo ágil de los lenguajes regulares, tanto conceptualmente (expresiones regulares, autómatas finitos) como a través de los programas que los usan (flex, egrep, sed...);
2. Manejo ágil de los lenguajes incontextuales, tanto conceptualmente (gramáticas incontextuales, analizadores de pila, análisis descendente y ascendente) como a través de los programas que los usan (bison, expat...)
3. Comprensión de las ampliaciones del análisis sintáctico mediante gramáticas de atributos, y su uso para gestionar tablas de símbolos y generar los resultados de traducción deseados.
4. Intuiciones adicionales: optimización de código, aplicaciones en lenguas humanas...

# Bibliografía (I)

## Disponible en la biblioteca

### Jeffrey D. Ullman y (algunos de) sus libros

Libro de teoría de la compilación (2 vols):

Aho, Ullman: Theory of Parsing, Translation, and Compiling, 1972

1969–1984: Varios libros por Ullman y cero, uno o dos de entre Aho y Hopcroft. Entre ellos, el primer "Dragón":

Aho, Ullman: Principles of Compiler Design, 1977

(dragonbook.stanford.edu)

1986–2006: Nuevas ediciones revisadas de los muchos libros, frecuentemente incluyendo nuevos autores; también algún que otro libro nuevo del todo. Entre ellos, los nuevos "dragones":

Aho, Sethi, Ullman: Compiler Design: Principles, Tools, and Techniques, 1986

Aho, Lam, Sethi, Ullman: Compilers: Principles, Tools, and Techniques, 2006

# Bibliografía (II)

## Disponible en la biblioteca u online


Mogensen, T.: "Basics of Compiler Design". http://www.diku.dk/~torbenm/Basics, también disponible en lulu.com (y en Amazon).

Appel, A. W.; Palsberg, J.: "Modern Compiler Implementation in Java" ("in C", "in ML"). Cambridge University Press 1998, 2002...

Grune, D.; Bal, H.; Jacobs, C; Langendoen, K.: "Modern Compiler Design". John Wiley & Sons 2000...

Wilhelm, R.; Maurer, D.: "Compiler Design". Addison-Wesley 1995, Pearson 2001...

Levine, J. R.; Mason, T.; Brown, D.: "Lex & Yacc". O'Reilly 1992, 1995...

Gálvez, S.; Mora, M.A.: "Compiladores con Lex/Yacc, JFlex/cup y JavaCC". lulu.com, 2005 (enlace desde la página "Compiladores" de wikipedia.es).

# Presentación Oral : lenguaje X

Nombre del estudiante: _____

| CATEGORÍA | 4 | 3 | 2 | 1 |
|---|---|---|---|---|
| Volumen | El volumen es lo suficientemente alto para ser escuchado por todos los miembros de la audiencia a través de toda la presentación. | El volumen es lo suficientemente alto para ser escuchado por todos los miembros de la audiencia al menos 90% del tiempo. | El volumen es lo suficientemente alto para ser escuchado por todos los miembros de la audiencia al menos el 80% del tiempo. | El volumen con frecuencia es muy débil para ser escuchado por todos los miembros de la audiencia. |
| Postura del Cuerpo y Contacto Visual | Tiene buena postura, se ve relajado y seguro de sí mismo. Establece contacto visual con todos en el salón durante la presentación. | Tiene buena postura y establece contacto visual con todos en el salón durante la presentación. | Algunas veces tiene buena postura y establece contacto visual. | Tiene mala postura y/o no mira a las personas durante la presentación. |
| Seguimiento del Tema | Se mantiene en el tema todo (100%) el tiempo. | Se mantiene en el tema la mayor parte (99-90%) del tiempo. | Se mantiene en el tema algunas veces (89%-75%). | Fue difícil decir cuál fue el tema. |
| Habla Claramente | Habla claramente y distintivamente todo (100-95%) el tiempo y no tiene mala pronunciación. | Habla claramente y distintivamente todo (100-95%) el tiempo, pero con una mala pronunciación. | Habla claramente y distintivamente la mayor parte (94-85%) del tiempo. No tiene mala pronunciación. | A menudo habla entre dientes o no se le puede entender o tiene mala pronunciación. |
| Límite-Tiempo | La duración de la presentación es adecuada. | La presentación es breve pero bastante completa. | La presentacón es bastante larga. | La presentación es muy breve e incompleta, o demasiado larga. |
| Vocabulario | Usa vocabulario apropiado para la audiencia. Aumenta el vocabulario de la audiencia definiendo las palabras que podrían ser nuevas para ésta. | Usa vocabulario apropiado para la audiencia. Incluye 1-2 palabras que podrían ser nuevas para la mayor parte de la audiencia, pero no las define. | Usa vocabulario apropiado para la audiencia. No incluye vocabulario que podría ser nuevo para la audiencia. | Usa varias (5 o más) palabras o frases que no son entendidas por la audiencia. |
| Contenido | Demuestra un completo entendimiento del tema. | Demuestra un buen entendimiento del tema. | Demuestra un buen entendimiento de partes del tema. | No parece entender muy bien el tema. |
| Tono | El tono usado expresa las emociones apropiadas. | El tono usado algunas veces no expresa las emociones apropiadas para el contenido. | El tono usado expresa emociones que no son apropiadas para el contenido. | El tono no fue usado para expresar las emociones. |
| Comprensión | El estudiante puede con precisión contestar casi todas las preguntas planteadas sobre el tema por sus compañeros de clase. | El estudiante puede con precisión contestar la mayoría de las preguntas plantedas sobre el tema por sus compañeros de clase. | El estudiante puede con precisión contestar unas pocas preguntas planteadas sobre el tema por sus compañeros de clase. | El estudiante no puede contestar las preguntas planteadas sobre el tema por sus compañeros de clase. |
| Entusiasmo | Expresiones faciles y lenguaje corporal generan un fuerte interés y entusiasmo sobre el tema en otros. | Expresiones faciales y lenguaje corporal algunas veces generan un fuerte interés y entusiasmo sobre el tema en otros. | Expresiones faciales y lenguaje corporal son usados para tratar de generar entusiasmo, pero parecen ser fingidos. | Muy poco uso de expresiones faciales o lenguaje corporal. No genera mucho interés en la forma de presentar el tema. |

# Planificación en Grupo

Nombre de los estudiantes: _____

| CATEGORIA | 4 | 3 | 2 | 1 |
|---|---|---|---|---|
| Plazo de Tiempo del Grupo | El grupo desarrolla un plazo de tiempo razonable y completo describiendo cuándo las diferentes partes del trabajo (por ejemplo, planeación, investigación, primer borrador, borrador final) estarían terminadas. Todos los estudiantes en el grupo pueden describir el plazo de tiempo usado. | El grupo desarrolla un plazo de tiempo que describe cuándo la mayoría de las partes estarían terminadas. Todos los estudiantes en el grupo pueden describir el plazo de tiempo usado. | El grupo desarrolla un plazo de tiempo que describe cuándo la mayoría de las partes estarían terminadas. La mayoría de los estudiantes en el grupo pueden describir el plazo de tiempo usado. | El grupo necesita la ayuda de un adulto para desarrollar un plazo de tiempo y/o varios estudiantes en el grupo no saben qué plazo de tiempo fue usado. |
| Delegación de Responsabilidad | Cada estudiante en el grupo puede explicar que información es necesaria para el grupo y qué información él o ella es responsable de localizar y cuándo es necesaria. | Cada estudiante en el grupo puede explicar qué información él o ella es responsable de localizar. | Cada estudiante en el grupo puede, con la ayuda de sus compañeros, explicar qué información él o ella es responsable de localizar. | Uno o más estudiantes en el grupo no pueden explicar qué información ellos son responsables de localizar. |
| Plan para la Organización de la Información | Los estudiantes tienen desarrollado un plan claro para organizar la información conforme ésta va siendo reunida. Todos los estudiantes pueden explicar el plan de organización de los descubrimientos investigados. | Los estudiantes tienen desarrollado un plan claro para organizar la información al final de la investigación. Todos los estudiantes pueden explicar este plan. | Los estudiantes tienen desarrollado un plan claro para organizar la información conforme ésta va siendo reunida. Todos los estudiantes pueden explicar la mayor parte de este plan. | Los estudiantes no tienen un plan claro para organizar la información y/o los estudiantes no pueden explicar su plan. |
| Calidad de las Fuentes | Los estudiantes identifican por lo menos 2 fuentes confiables e interesantes de información para cada una de sus ideas o preguntas. | Los estudiantes identifican por lo menos 2 fuentes confiables de información para cada una de sus ideas o preguntas. | Los estudiantes, con ayuda de un adulto, identifican por lo menos 2 fuentes confiables de información para cada una de sus ideas o preguntas. | Los estudiantes, con bastante ayuda de un adulto, identifican por lo menos 2 fuentes confiables de información para cada una de sus ideas o preguntas. |