

# INFORME

## Autómatas, Teoría de Lenguajes y Compiladores

**Simple Keyboard Language**

#### Santiago Burgos (5), Gabriel Silvatici (53122).





**Introducción**

El objetivo de este trabajo práctico fue implementar diferentes tipos de IPC en un sistema POSIX. Concretamente la tarea realizada consiste en repartir un grupo de archivos por parte de un proceso llamado aplicación (applicationProcess) a procesos esclavos (slaveProcess) que deben retornar los nombres de los archivos acompañados de los hashes de los mismos. El hash es obtenido mediante el comando md5sum. Por último, el proceso aplicación guarda los datos recibidos en un buffer compartido con un proceso vista (viewProcess). El proceso vista cuando es ejecutado muestra en salida estándar los datos guardados en el buffer.

En este informe se explicarán las estructuras e IPCs utilizados, problemas encontrados en la implementación y sus respectivas soluciones. También se detallarán la bibliografía utilizada, fragmentos de código utilizados como referencia y las instrucciones de instalación y ejecución.

### Implementación

Se utilizo lex y yacc para la definición de la gramática, y se obtiene un programa en código Java, para realizar la traducción se utilizó una pila para ir resolviendo las llamadas hasta obtener una función luego esa función se la añade a un array de funciones ya traducidas a código java.

Se tiene una array con los nombres de las funciones y con las variables para evitar funciones repetidas, falta de función main, y declaración repetida de variables o nombrar variables inexistentes.

Gramatica

Tipos de datos:

* Integer

“int ‘name’ eq 0,”

* String

“str ‘name’ eq .Soy un string,”

La igualdad se representa con “eq”

El fin de linea se representa con “,”

Constante

“con ‘name’ eq .Soy una constante”

1



Operadores

“add” (+), “sub” (-), “mul” (\*), “div” (/), “mod” (%), “gre” (>),

“les” (<), “and” (&&), “or” (||), “not” (!), “cmp” (==).

Programa

* skl

fun ‘name’..

..

fun main...

..

Se usa “skl” para indicar inicio de programa y luego se declaran las funciones.

* Llamada a funcion

“go ‘functionname’,”

* Funciones

fun ‘name’ ..

..

Se usa “..” para indicar comienzo y final del cuerpo de la funcion.

Para que el programa funcione necesita una funcion main (fun main..)

* If

if , ‘condicion’ , ..

..

Se usa “,” para comienzo y final de la condicion, “..” para el cuerpo

* While:

whi , condicion , ..

..

* For:

for , ‘name’ eq ‘value’ to ‘target’ ..

..

Donde name puede ser una variable nueva o ya declarada, value y target pueden ser un numero o el nombre de una variable ya declarada.

For incrementa name desde value hasta target.

Ejemplo de condicion: “ , , ‘value1’ cmp 10, and ,value2 cmp 10, , “

seria equivalente a

“(( ‘value1’ == 10) && ( ‘value2’ == 10))” en C.

2



* Imprimir en pantalla:

“put .Hola mundo,”

“put ‘name’”

* Leer de pantalla:

“get ‘name’”

Asigna a la variable ‘name’ el valor de get. Solo se admite lecturas de integer y el integer ‘name; debe estar previamente declarado.

Limitaciones a considerar

Las funciones son de tipo void y sin argumentos, una funcion no puede ser llamada si no fue declarada antes. Las variables no tienen un valor por defecto, deben setearse al declararlas. Se podria expandir la cantidad de tipos y explotar ventajas de las librerias de java aunque al pensarse en un lenguaje con pocos caracteres especiales habria que expandir mucho la cantidad de palabras reservadas.

3