|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Manejo de la Recursividad mediante Paso de Control y Paso de Parámetros**  [www.fgalindosoria.com/informatica/properties/recursion/art/recursividad\_control\_y\_parametros.pdf](http://www.fgalindosoria.com/informatica/properties/recursion/art/recursividad_control_y_parametros.pdf)  **Fernando Galindo Soria**  <http://www.fgalindosoria.com/>  Ciudad de México, 1974 - 1975  Capturado en Tenayuca, Ciudad de México, el 29 de Abril del 2017  Los procesos recursivos se pueden realizar secuencialmente, en paralelo, síncronamente, asíncronamente, pero en general son procesos en paralelo asíncronos. Uno de los problemas que tenemos para ejecutar este tipo de procesos es que, la mayoría de las maquinas cuentan con un solo procesador o pocos procesadores, por lo que al final, las maquinas donde se ejecutan son secuenciales no recursivas, en ese caso se tiene que simular en la maquina secuencial el proceso recursivo.  Para poder entender cómo se realiza el manejo de la recursividad es conveniente entender cómo se realizan otros procesos, como el paso de control y el paso de parámetros, para lo cual, mostraremos como se realizan estos procesos desde el nivel del código de la maquina hasta el manejo de interrupciones y procesos en general (la recursividad, desde el punto de vista de cómo se procesa, es un caso particular del manejo de procesos), por lo que se verán:  El manejo del PC (Program Counter). (Tema de la Arquitectura de Computadoras)  El Manejo de saltos (Go To). (Tema de los Compiladores)  El Manejo de Interrupciones. (Tema de los Sistemas Operativos)  El Manejo de Rutinas. (Tema de los Compiladores)  El Manejo de la Recursividad. (Tema de las Matemáticas, programación)  Prácticamente desde las primeras computadoras digitales, se empezaron a desarrollar procesos que llaman o son llamados desde otros procesos. Por ejemplo, cuando un programa llama a otro proceso que calcula el seno o el coseno o manda imprimir algo. Estas son cosas que se realizan comúnmente dentro de las computadoras y que no son triviales computacionalmente, por lo que desde el principio de la programación se establecieron como módulos (procesos, aplicaciones, rutinas, funciones, interrupciones, bloques, etc.) que pueden ser llamados desde otros módulos, a continuación, mostraremos como se ha realizado este tipo de actividad.  Los dos primeros problemas a los que nos enfrentamos es el paso de control y el paso de parámetros, estos son problemas que se presentan en toda la programación (aunque en general, la gente no se da cuenta, ya que las maquinas desde el principio traen integradas herramientas para realizarlos o ya se encuentran en las capas básicas de la programación de los sistemas). El *Paso de Control* engloba los mecanismos que le indica a la máquina, quien tiene el control de los procesos o cual es la siguiente instrucción a ejecutar. El *Paso de Parámetros* engloba los mecanismos que le indican a la máquina con que datos se va a ejecutar y como se preservan datos de otros procesos (lo que se conoce como el *estado del proceso*).  Para entender estos mecanismos, como primer paso comenzaremos viendo un poco de la forma como las maquinas manejan y deciden cual es la siguiente instrucción a ejecutar.  **Manejo del PC (Program Counter)**  (Arquitectura de Computadoras)  Mínimo de Arquitectura de Computadoras  Si se estudia a nivel básico una computadora, se ve que tiene una serie de circuitos básicos (aunque no vamos a profundizar, es conveniente saber que existen para poder entender los artículos sobre el tema), como la ALU (Unidad Aritmético Lógica), una serie de registros de trabajo (Ri) donde se almacenan los datos durante las etapas en las que se están procesando, el más común e importante de estos circuitos se conoce como Acumulador (AC), también se tienen los puertos, que son circuitos que permiten conectar el núcleo de la computadora con los dispositivos que lo rodean (como el teclado, mouse, pantalla, discos, memorias, redes, etc.), se tienen varios canales de comunicación (conocidos como bus) entre los diferentes circuitos, como el bus de datos y el bus de direcciones.  Además, en las maquinas digitales (recuérdese que existe una gran cantidad de máquinas de tipo analógico, tal vez mas que digitales) se cuenta con diferentes tipos de almacenamiento de información, como memoria, discos, USB, etc. En general todos los dispositivos de almacenamiento cumplen con una serie de estándares para poder ser compatibles entre sí y con los otros dispositivos. Entre estos estándares se tiene lo que se conoce como *palabra* y tamaño de la palabra, la palabra es la unidad básica de almacenamiento y actualmente su tamaño es de 8, 16, 32 o 64 bits, dependiendo de la maquina (actualmente casi todas las maquinas tiene esos tamaños, pero han existido maquinas con 7, 9, 12, 72, etc. bits por palabra), prácticamente todas las maquinas actuales son binarias, aunque han existido y existen máquinas que no son binarias, máquinas de base 3, decimales, base 12, etc.  En general las palabras actuales son cadenas de bits, o sea que lo único que se almacena son 0´s y 1´s, pero esas cadenas de 0´s y 1´s se interpretan de múltiples formas, dependiendo de los circuitos que los interpretan, por ejemplo una palabra puede interpretarse como un numero o dato, como una dirección donde se almacena algo, como una instrucción u orden sobre el sistema (como las instrucciones de suma, resta, salto, leer, etc.), aun mas en algunos casos en una palabra puede estar compuesta con una combinación de componentes, por ejemplo, dentro de la palabra unos bits pueden indicar que operación se debe realizar, y otros bits, pueden ser la direcciones donde están los bis sobre los que se realiza la operación. En general los formatos de las instrucciones pueden ser:  dato  Operación  Operación, valor del dato  Operación, dirección del dato  Operación, dirección del dato 1, dirección del dato 2  Operación, dirección del dato 1, dirección del dato 2, dirección del dato3  (En este caso la operación se realiza entre los datos 1 y 2 y el resultado se almacena en la dirección del dato3)  Dirección  (En este caso la palabra tiene almacenada una dirección (por ejemplo, la dirección a la que debe saltar la máquina para seguir ejecutándose))  Aún más, existen maquinas, donde un dato, instrucción o dirección ocupan varias palabras o donde en una sola palabra pueden almacenarse varias instrucciones  Palabra (informática)  <https://es.wikipedia.org/wiki/Palabra_(inform%C3%A1tica)>  Unidades de información  <https://es.wikipedia.org/wiki/Unidades_de_informaci%C3%B3n>  Paso de Control usando el PC (Program Counter).  Desde el surgimiento de las primeras computadoras digitales se tuvo la necesidad de saber dónde estaban almacenados los datos e instrucciones de un programa, la solución fue genial y extremadamente simple, dentro del hardware de la maquina se introdujo un circuito conocido como IP (Instruction Pointer) o PC (Program Counter) (Comentario: este nombre no tiene nada que ver con el termino PC de las Computadoras Personales), el PC es un circuito donde se almacena la dirección donde se encuentra la siguiente instrucción, o sea que, cuando la computadora ejecuta una instrucción (por ejemplo una suma o una resta), sabe dónde se encuentra la siguiente instrucción a ejecutar gracias a que el PC se lo está diciendo.  Un pequeño detalle que ocasiona este tipo de solución es que, las maquinas siempre toman la siguiente instrucción a la que apunta el PC, la ejecutan, toman la siguiente instrucción, la ejecutan y así permanentemente, lo que hace que, las computadoras siempre estén funcionando, desde que se prenden hasta que se apagan, pero puede llegar a ocurrir que la maquina no tiene procesos en espera, por lo que, desde las primeras computadoras se implementó un proceso conocido como INLE que vendría siendo el proceso cero del sistema, en algunos casos el proceso INLE realmente no hace algo (muchas maquinas tienen la instrucción NULL que no hace algo pero consume tiempo), pero muchísimas maquinas se ponen a resolver algún problema no indispensable, pero que, en su momento puede servir para algo, como por ejemplo el cálculo de π o algo parecido, otras realizan algún proceso de mantenimiento, etc.  Contador de programa  <https://es.wikipedia.org/wiki/Contador_de_programa>  Ciclo de fetch o ciclo de instrucción  **C**ICLO DE FETCH  <https://es.scribd.com/doc/22376834/Ciclo-de-Fetch>  Ciclo de instrucción  <https://es.wikipedia.org/wiki/Ciclo_de_instrucci%C3%B3n>  Ciclo De Instrucción  <https://www.youtube.com/watch?v=OpVt-dG-KgE>  L3 2 Funcionamiento del procesador: ejecución de instrucciones.  <https://www.youtube.com/watch?v=GgjFgPWGIUM>  **Manejo de saltos (Go To)**  (Compiladores)  Entonces, cuando se está ejecutando un programa, siempre sabemos cuál es la siguiente instrucción a ejecutar gracias al PC (Program Counter), pero es aquí donde la situación se empieza a complicar, ya que dentro de la ejecución de un programa a veces queremos ejecutar una instrucción que se encuentra en otro lugar, por ejemplo, a veces queremos ejecutar un ciclo (o sea repetir una secuencia de instrucciones), como, por ejemplo:  Mientras i>1  i= i-1  imprime i  Regresa  Siguiente instrucción  O sea que:  Mientras i sea mayor que 1, réstale 1 a i, imprime el valor de i, regresa a repetir el ciclo.  O sea que, ejecuta una secuencia de instrucciones y luego tiene que regresar al principio, para repetir el ciclo y así hasta que se cumple la condición. En este caso llega un momento en que la maquina ya no debe ejecutar la siguiente instrucción, sino que debe de regresar a ejecutar una instrucción previa. Lo que significa que, debe existir algún mecanismo para que el PC ya no apunte a la siguiente instrucción, sino que apunte a otra dirección.  Esto se logra introduciendo lo que se conoce como Etiqueta y Vector de etiquetas, de tal manera que cuando el compilador (que es el que genera el código que debe ejecutar la maquina) detecta la instrucción Mientras, lo primero que hace es inventar un nombre (al que se conoce como etiqueta) y almacena la etiqueta que invento y la dirección donde se presenta la instrucción Mientras, en una tabla conocida como vector de etiquetas. Y cuando se termina el ciclo se genera un código para que en el PC se cambie el valor de la siguiente instrucción que se debería ejecutar, por la dirección donde empieza el Mientras.  Mientras i>1  i= i-1  imprime i  Regresa  Siguiente instrucción  **E0** Mientras i>1  i= i-1  imprime i  Regresa  **E1** Siguiente instrucción  …..  …..  E1 Dirección de la Siguiente instrucción  E0 Dirección del Mientras  1  2  **1** inventa un nombre (al que se conoce como etiqueta)  **2** almacena la etiqueta que invento y la dirección donde se presenta la instrucción Mientras, en una tabla conocida como vector de etiquetas.  **3** cuando se termina el ciclo se genera un código para que en el PC se cambie el valor de la siguiente instrucción que se debería ejecutar, por la dirección donde empieza el Mientras.  Ahora bien, la mayoría de las computadoras no tiene dentro de sus instrucciones básicas (instrucciones de máquina) a la instrucción Mientras, realmente la instrucción Mientras, se sustituye durante la etapa de compilación o de interpretación por varias instrucciones. Por ejemplo, el código anterior quedaría representado por un conjunto de instrucciones parecidas a lo siguiente  E0 Si i <= 1 Go To E1  i= i-1  imprime i  Go To E0  E1 Siguiente instrucción  Si i es menor o igual que 1 salta a la instrucción que tiene la dirección (etiqueta) E1,  sino decrementa i, imprime el valor de i.  Regresas a la instrucción que compara a i con el 1 (etiqueta E0)  O sea que las instrucciones del tipo Mientras (while), se transforman dentro de la computadora.  Primero se recuerda de alguna manera en que posición se encuentra la instrucción Mientras, en este caso la forma de recordarlo fue marcándolo con una etiqueta (E0).  Se invierte la comparación (obsérvese que en el código original se repetía el ciclo Mientras i > 1, y en el código interno se pregunta lo contrario o sea que si i <= 1)  Si i <= 1 salta (Go To) a la primera instrucción que esta después del ciclo (que en este caso lo etiquetamos como E1).  OJO, obsérvese que estamos usando un Go To, y desde que se introdujo la programación estructurada, explícitamente se prohibieron los Go To, esto fue porque, el Go To mal usado, daba como resultado la generación de lo que se conoce como programación espagueti, o sea códigos llenos de saltos en forma indiscriminada y muy difíciles de entender, pero resulta que, a nivel de máquina, el salto o Go To es una de las instrucciones básicas, ya que es muy fácil de implementar, como se verá más adelante.  Por otro lado, existen algoritmos de equivalencia de las instrucciones iterativas y de decisión, como el while, do wile, e if then else, a códigos muy directos dentro de las maquinas, mediante la introducción de etiquetas y de Go To´s que no generan código espagueti, como se mostrara también a continuación.  Si no se cumple la condición interna i <= 1 (o sea si i > 1), se ejecuta el código del ciclo original (aquí es donde se ve lo bueno de haber invertido la condición, porque por la misma estructura del programa se termina ejecutando el código del ciclo cuando i > 1)  Mediante otro Go To se regresa a verificar la condición y en su caso a repetir el ciclo  O sea que, dentro de la mayoría de las computadoras no existen internamente las instrucciones clásicas de iteración y decisión, por lo que, se tienen que sustituir por etiquetas y Go To´s (esto lo realizan los compiladores y los interpretes de los lenguajes) como se muestra a continuación:  Código Original  Mientras condición  ejecuta ciclo  Representación mediante etiquetas y Go To (las flechas indican los Go To)  V  E0 ~condición ejecuta ciclo E1  F  Código Original  If condición then opción verdadera else opción falsa siguiente instrucción  V  F  condición opción verdadera E1 opción falsa E2 siguiente instrucción  Ya que se tiene el código en termino de Go To´s y etiquetas, implementarlo en términos del lenguaje y arquitectura de las maquinas es fácil.  Como se recuerda, cada instrucción de la maquina esta almacenada en alguna dirección física, por lo que, cuando se detecta que se tiene que registrar una etiqueta, lo que se hace es almacena la dirección en una tabla de etiquetas. OJO aquí es muy importante darnos cuenta que se están inventando localidades auxiliares dentro de la máquina (conforme se avanza en el manejo interno se ve que cada vez es más necesario crear códigos y espacios de apoyo).  Para implementar los Go To´s recordamos que el PC de la maquina apunta a la siguiente instrucción a ejecutar, por lo que, *hacer un Go To a alguna dirección especifica consiste simplemente en sustituir el valor que tiene el PC, por el valor de la dirección específica a la que se quiere saltar*.  **Manejo de Interrupciones**  (Sistemas Operativos)  Como se vio, el salto a alguna posición específica, es un proceso muy simple para el hardware de las máquinas, ya que solo consiste en poner la dirección a la que se quiere saltar en el PC, a continuación, se mostrará cómo se usa este tipo de herramientas para múltiples propósitos.  El primer ejemplo se da en lo que se conoce como manejo de interrupciones, los sistemas de cómputo están construidos de tal manera que, sea relativamente fácil interrumpir un proceso, ejecutar otro proceso y regresar al punto donde se realizó la interrupción, esto es muy práctico, ya que permite, por ejemplo, interrumpir un proceso dentro de las computadoras, cuando ocurre alguna situación que requiere atención, por ejemplo, cada que movemos el mouse o tecleamos algo en la máquina, es necesario que la máquina le preste atención (como se comentó desde el principio, estamos suponiendo que nuestra máquina es secuencial o tiene pocos procesadores, o sea que, tarde o temprano un procesador tiene que interrumpir lo que está haciendo, para prestar atención a algo).    Un proceso sin interrupciones  Rutina 0 R0  Un proceso con una interrupción  Rutina 0 R0  Rutina 1 R1  Punto de Interrupción  Llamado a la rutina de interrupción  Regreso de la rutina de interrupción  La mayoría de las computadoras tienen mecanismos por hardware para el manejo de interrupciones, pero existen lo que se conoce como interrupciones lógicas que requieren programación explicita para su manejo, en general las interrupciones lógicas y físicas se manejan en forma similar.  Para el manejo de las interrupciones, primero se tiene que detectar la interrupción, después se debe saber qué hacer cuando se detecta esa interrupción, para lo que se tiene lo que se conoce como *vector de interrupciones*, que es una tabla dentro de la computadora, donde a cada tipo de interrupción, se le asocia la dirección donde está el código que se tiene que ejecutar cuando se detecta esa interrupción, además las interrupciones de hardware, como por ejemplo el movimiento del mouse están cableadas (o sea se tiene un circuito que conecta el puerto (entrada a la que está conectado el aparato que envía la interrupción, por ejemplo el mouse) al manejador de interrupciones), cuando se mueve el mouse se genera una señal que le llega al manejador de interrupciones, esté interrumpe el proceso que se está ejecutando, almacena en algún lugar la dirección del programa interrumpido, toma la dirección de la rutina que realizo la interrupción (la toma del vector de interrupciones) y la guarda en el PC, las rutinas de interrupción terminan con una especie de go to (conocido como RTI (Retorno de Interrupción)) que lo que hace es recuperar la dirección del proceso interrumpido y regresarle el control a esté.  Hasta aquí todo se ve bien pero se tiene dos detalles fundamentales, el primero es que cuando una rutina R1 interrumpe a otra R0, también puede ser interrumpida por otra rutina R2, y está por otra R3 y así se puede crear una cadena de rutinas que han sido interrumpidas y están pendientes de que termine la que las interrumpió, para poder continuar, entonces el problema es, como sabe la maquina a donde regresar después de cada interrupción, la solución que se usa es que las direcciones de regreso de las rutinas interrumpidas no se almacenan en una variable, se almacenan en una pila (recuérdese que una pila es una estructura de datos, donde el último dato almacenado es el primero en recuperarse) de tal manera que la última rutina de interrupción Rn cuando termina recupera la dirección de la rutina que interrumpió Rn-1 y esta cuando terminan recupera la dirección donde debe continuar Rn-2, y así sucesivamente hasta que R2 le regresa el control a R1 y está a R0, continuando el proceso original donde se quedó.  Hasta este punto se ha tratado un problema que se conoce como *Paso de Control* (o sea como se le indica a la maquina quien tiene el control de los procesos o cual es la siguiente instrucción a ejecutar), pero el siguiente problema que veremos corresponde a lo que se conoce como *Paso de Parámetros* (Como se le indica a la maquina con que datos se va a ejecutar. Como se preservan datos de otros procesos que fueron interrumpidos, que se necesita evitar que se pierdan y que en su momento se puedan usar cuando se necesiten).  Específicamente, cuando una rutina se interrumpe estaba realizando algo y por tal motivo haciendo cálculos, tomando decisiones, manejando datos, por lo que al ser interrumpida se requiere guardar y recordar de alguna forma lo que se conoce como el *estado del proceso*, Entonces los sistemas no solo deben preservar la dirección de las rutinas interrumpidas, sino también el estado de cada uno de los procesos interrumpidos.  Manejo de Interrupciones  Rutina 0 R0  Punto de Interrupción 1  manejador de interrupciones  1  Interrumpe el proceso actual  Guarda la dirección de regreso Ri y el estado de la rutina Ei en la pila  Dispositivo o rutina que envía un requerimiento de interrupción (por ejemplo, el mouse)  Puerto  Rn E0n  Rn-1 E0n-1  ……….  R2 E2  R1 E1  R0 E0  PC  Ri  Interrupción 0  Interrupción 1  Interrupción 2  Interrupción m-1  Interrupción m  3 Regreso de la rutina de interrupción  2 Llamado a la rutina de interrupción  Rutina 1 R1  2  Toma la dirección de la rutina de interrupción del vector de interrupciones y la coloca en el PC  Vector de Interrupciones  Pila  **3** Regreso de la rutina de interrupción  Toma del último elemento de la pila la dirección de regreso Ri y el estado de la rutina Ei,  Restablece los valores del estado de la rutina en las posiciones donde estaban antes de la interrupción  y coloca la dirección de regreso en el PC    Interrupción  <https://es.wikipedia.org/wiki/Interrupci%C3%B3n>  **Manejo de Rutinas**  (Compiladores)  Existe una gran cantidad de rutinas que se usan cotidianamente y que no tiene objeto programar cada que se requiere, como por ejemplo, calcular el valor del seno o del coseno o mandar imprimir un texto, por lo que, por un lado, los sistemas tienen una gran cantidad de rutinas (procesos, subrutinas, funciones, objetos, aplicaciones, módulos, métodos, etc.) ya preprogramados, y por otro lado, los mismos lenguajes de programación proporcionan mecanismos para que podamos crear nuestras propias rutinas.  Módulo 6  Módulo 4  Módulo 2  Módulo 1  Módulo 0  Módulo 5  Módulo 6  Módulo 3  Para el manejo interno de las rutinas cada rutina se almacena en algún lado del sistema y se tienen tablas donde se almacena el nombre de la rutina y la dirección donde esta físicamente almacenada  Paso de Control  Cuando se manejan las rutinas, el primer problema es el Paso de Control, que en este caso consiste en que, cuando un proceso llama a otro, lo primero que se hace es recordar de alguna forma donde se realizó el llamado, para poder regresar al mismo punto ya que se realizó el proceso, y como siguiente paso se coloca la dirección del salto (de la rutina que se quiere ejecutar) en el PC (Program Counter). Por otro lado, cuando termina su proceso la rutina que fue llamada, regresa el control a la rutina que la llamo, colocando la dirección de regreso (la toma del lugar donde la guardo la rutina que llamo) en el PC. Ahora bien, como la rutina llamada puede llamar a otra y está a otra y así sucesivamente, normalmente las direcciones de regreso se guardan en una pila.  Paso de parámetros  (Parámetros globales y locales)  El paso de parámetros entre las múltiples rutinas que se están llamando, puede ser de muchas formas, a veces se pasa directamente el valor de un dato y en otras se pasa el nombre de la variable donde está el dato. Por otro lado, cuando se está programando, en algunos casos es importante que todo el sistema tenga acceso a ciertos datos específicos, independientemente de que sean procesos principales, rutinas o rutinas dentro de rutinas, a estas variables se les conoce como *variables globales*, en otros casos es importante que solo una rutina especifica se comunique con alguna otra, a estos parámetros se les conoce como *variables o parámetros locales*, entonces dependiendo del tipo de parámetro que se quiere manejar, se usan diferentes técnicas, pero en general se tienen 4 casos, se envía explícitamente el dato o la dirección donde está el dato, se da acceso global o local.   |  |  |  | | --- | --- | --- | | Paso de parámetros | Global | Local | | Dato | Los usan todas las rutinas, pero no modifican su valor | Solamente los usan las rutinas a las que se les envía, pero no modifican su valor | | Dirección | Los usan todas las rutinas, y pueden modificar el valor de la variable | Solamente los usan las rutinas a las que se les envía y pueden modificar el valor de la variable |   El manejo de parámetros globales es relativamente simple ya que únicamente se crea una tabla donde se almacenan la información (indicando si son datos o direcciones) y a la cual tienen acceso todas las rutinas.  El manejo de parámetros locales consiste básicamente en asociar al llamado de las rutinas la lista de variables a pasar, indicando si se pasan datos o variables, entonces la rutina toma esos valores y si son variables (direcciones) cuando regrese del llamado debe asegurarse que las variables almacenen los valores resultantes del proceso.  El problema surge cuando se maneja lo que llamaremos parámetros globales locales.    Por ejemplo, en la siguiente imagen, en el módulo 0 se declaran 2 variables globales x1, x2.  Pero en módulo 2 se declaran 3 variables de tipo global y1, y2, y3.  Entonces las variables x1 y x2 son globales para todos los módulos, pero y1, y2, y3 son globales solo para los módulos 2, 4, 5 y 7  Módulo 0  Módulo 1  Módulo 6  Módulo 5  Módulo 4  Módulo 3  Módulo 2  Módulo 7  x1, x2  y1, y2, y3    **Manejo de Recursividad**  (Matemáticas, programación)  El manejo de la recursividad es prácticamente igual al manejo de cualquier llamado de un módulo o rutina por otra, solo se tiene que tener cuidado con distinguir entre los diferentes llamados entre los módulos  Módulo 0  Módulo 0  Módulo 0  Módulo 0  ……..  Por ejemplo *el Módulo 0 se llama en forma recursiva múltiples veces, pero desde el punto de vista de la máquina, cada llamado se puede ver como una instancia independiente*, ya que en cada llamado, la maquina almacena en la pila la dirección de regreso y coloca en el PC la dirección a donde salta (que en este caso la dirección a donde salta sea la misma a la que salto la vez anterior es indiferente para la máquina, ya que la primera vez que salto almaceno en la pila de saltos la dirección de donde salto y la segunda vez almacena la misma dirección del salto, pero *la almaceno en una posición de pila diferente*, o sea que cada llamado se puede ver como una instancia diferente)  Por facilidad mental distinguiremos cada llamado recursivo o instancia mediante algún mecanismo (por ejemplo, mediante un subíndice)  Módulo 0  Instancia 0  ……..  Módulo 0  Instancia 1  Módulo 0  Instancia 2  Módulo 0  Instancia n  Pila de Control  Almacena la dirección de regreso de cada llamado  Módulo 0 Instancia n  …………  …………  Módulo 0 Instancia 2  Módulo 0 Instancia 1  Módulo 0 Instancia 0  O sea que, realmente para la maquina NO se llama a la misma rutina ni las ve como la misma rutina (esto es fundamental para entender muchos procesos recursivos) y si se quiere llegar a un compromiso son DIFERENTES instancias de la MISMA rutina.  Lo mismo pasa con el paso de los parámetros locales, cuando llama a la rutina almacena los parámetros locales en una posición (manejada por la pila), y la siguiente vez que llama almacena los parámetros locales en otra dirección, ya que la maquina ve cada llamado, como un llamado a una rutina diferente o sea que para la maquina (y para nosotros) cada llamado recursivo, envía y recibe parámetros completamente diferentes de cualquier otro llamado recursivo.  Ejemplo  El viaje en el tiempo es un proceso recursivo  [www.fgalindosoria.com/notas\_de\_investigacion/el\_viaje\_en\_el\_tiempo\_es\_un\_proceso\_recursivo/el\_viaje\_en\_el\_tiempo\_es\_un\_proceso\_recursivo.pdf](http://www.fgalindosoria.com/notas_de_investigacion/el_viaje_en_el_tiempo_es_un_proceso_recursivo/el_viaje_en_el_tiempo_es_un_proceso_recursivo.pdf)  Fernando Galindo Soria  [www.fgalindosoria.com](http://www.fgalindosoria.com/).. ...[fgalindo@ipn.mx](mailto:fgalindo@ipn.mx).. ..[Red de Desarrollo REDI](http://www.fgalindosoria.com/redi/)  Tenayuca, Ciudad de México, 3 de Agosto del 2014  Es similar a como se maneja la recursividad en una computadora secuencial  Ver paso de control y paso de parámetros  [www.fgalindosoria.com/informatica/properties/recursion/art/recursividad\_control\_y\_parametros.pdf](http://www.fgalindosoria.com/informatica/properties/recursion/art/recursividad_control_y_parametros.pdf)  Cada que el proceso X se llama recursivamente se crea una nueva instancia o sea que la primera vez que se llama es la instancia X0 pero la siguiente es la instancia X1 y asi sucesivamente  Todo eso se maneja con pilas  Cuando A viaja al pasado por ejemplo una semana tiene una edad t, por lo que cuando llega al pasado llega con la edad t, pero todo lo que puede encontrar incluyendo a si mismo tiene una edad de t-1s |