Índice de contenido

Hipótesis y Aclaraciones

Problemas relevantes

◦ *Expansión del carácter "*" dentro de las ""*

Este problema se presentó al utilizar un for para recorrer los archivos del directorio. Al incluir el carácter "*" dentro de las "", el for toma como un único elemento la lista de los archivos contenidos separada por los espacios. Esto se solucionó colocando el carácter "*" fuera de las "".

◦ *Espacios en blanco en las rutas*

Al colocar espacios en las rutas de los directorios sobre los cuales se ejecutaba el programa, se observó que se debía "proteger" entre "" las rutas que utilizaban la variable de entorno $grupo, para evitar que los mismos produjeran errores a lo largo de la ejecución de cada uno de los comandos.

◦ *Espacios en blanco en los campos de los registros*

Al momento de probar el comando invreci, utilizando espacios en los registros, se presentó el problema de que el comportamiento del comando grep no era el esperado debido a los mismos. Por lo tanto, se procedió a reemplazar los espacios de los campos de los registros por el carácter "_", obteniéndose de este modo, el comportamiento esperado del grep.

◦ *Captura de señales para terminar un proceso*

Al desarrollar las funciones de start y stopinvonio para detener el proceso demonio, se intentó capturar la señal de kill. Luego de investigar, se observó que no era posible realizarlo de ese modo, dado que la señal de kill(9) no puede capturarse. Por lo tanto, se decidió capturar la señal de SIGTERM(15), y de este modo finalizar correctamente el comando invonio.

◦ *Problema con un flag del comando sed*

Inicialmente, para darle el formato correspondiente a los registros, se intentó utilizar el flag -i del comando sed. Al observar que no se lograba el comportamiento esperado del mismo, se decidió utilizar una variable auxiliar para modificar el formato y luego grabar el contenido de dicha variable en el archivo destino.

◦ *Error en la fórmula del cálculo de cumplimiento del enunciado*

Al desarrollar el comando occtrl utilizando la fórmula de cumplimiento dada en el enunciado se detectó un error. Al ejecutar el comando, con una orden de compra en la cual se esperaban, por ejemplo, 5(cinco) artículos y se habían recibido, por ejemplo, 0(cero); el resultado observado de la fórmula era el 100% de cumplimiento. Esto es claramente un error, dado que en el ejemplo planteado el grado de cumplimiento esperado sería del 0%. Dado esto, se decidió invertir el orden de la fórmula.

○ *Problema para invocar el invreci sólo si no estaba ya ejecutándose*

Al tratar de invocar el comando invreci desde el comando invonio, se debió validar que dicho comando no este en ese momento en ejecución. Esto presentó algunos problemas, como por ejemplo, el no poder validarlo y que se invoque infinitas veces el proceso invreci sin importar si se encontraba ejecutándose o no. Para solucionarlo, se procedió a validarlo utilizando el comando ps y el flag eo pid (para obtener sólo los pid's de los procesos ejecutándose) y luego un grep con el PID del invreci que se lanzó anteriormente; observar que al lanzarse el comando invreci siempre se guarda el PID con el que se ejecutó.

○ *Finalización del último campo del registro con ";"*

Al desarrollar el comando de validación de registros (invreci) se presentó el problema de no saber si el último campo de los registros finalizaba con un ";" (al igual que el resto de los campos). Para solucionar este problema, se tomó como convención que el mismo no finaliza con ";".

○ *Parseo de parámetros de invocación del comando occtrl*

Al desarrollar este comando se presentó el problema de parsear los parámetros para la invocación del mismo. Este problema se presentó a causa de que la cantidad de parámetros es muy variable, con muchas combinaciones posibles y aceptándose también, la invocación del mismo sin parámetros (utilizando parámetros por defecto). Esto se solucionó estableciendo como convención que primero debe pasarse como parámetro la salida y luego la cantidad de ordenes de compra a procesar(rango,simple,todas,etc). También, para simplificar la lectura e interpretación del código perl se optó por utilizar el módulo Switch y de este modo evitar anidar varios if's de modo sucesivo.

○ *Error en la fórmula de pendientes en occtrl del enunciado*

Al realizar el cálculo de pendientes, según la fórmula del enunciado, se detectó un error en la misma. Dado que, en esta, se calcula el pendiente como la cantidad total menos la cantidad remanente; lo cual es a nuestro entender incorrecto, dado que el pendiente es directamente la cantidad remanente de artículos.

Archivo README: Instructivo de Instalación

Diagrama de Procesos

- ***Diagrama de Proceso: Comando inovio***

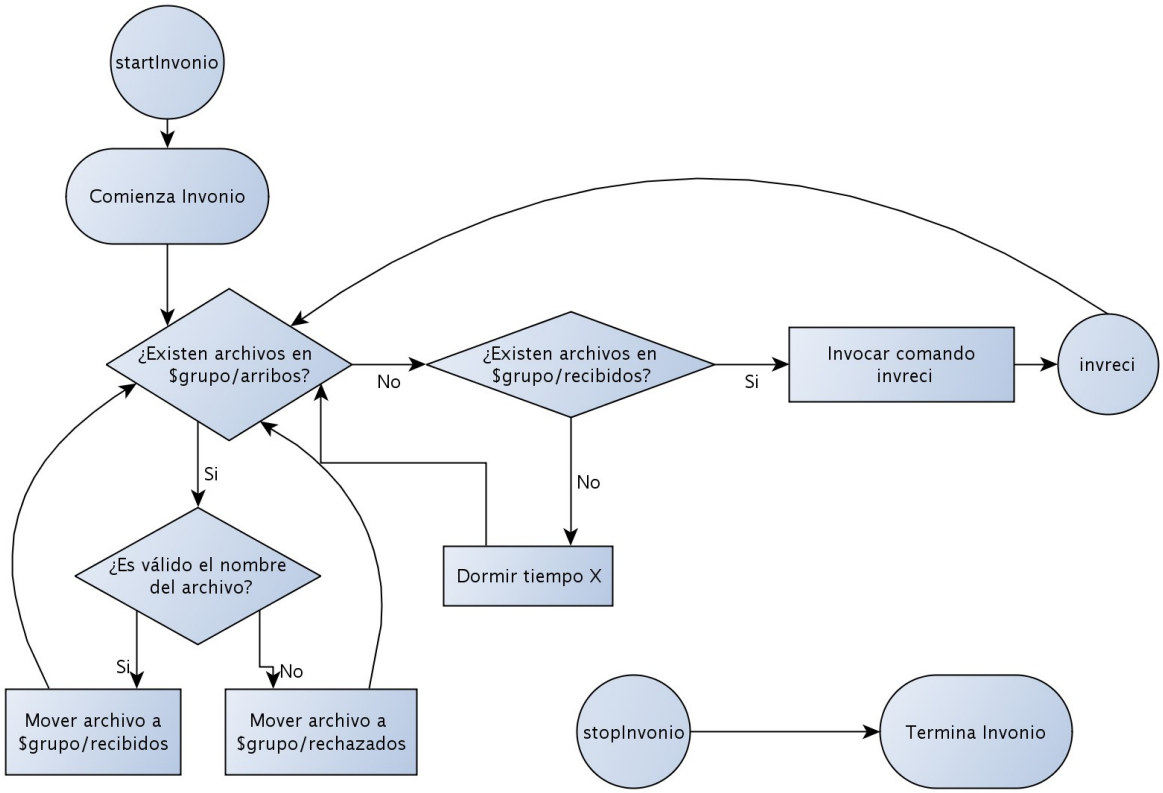

Diagrama de Proceso 1: Comando Invonio

- ***Diagrama de Proceso: Comando invreci***

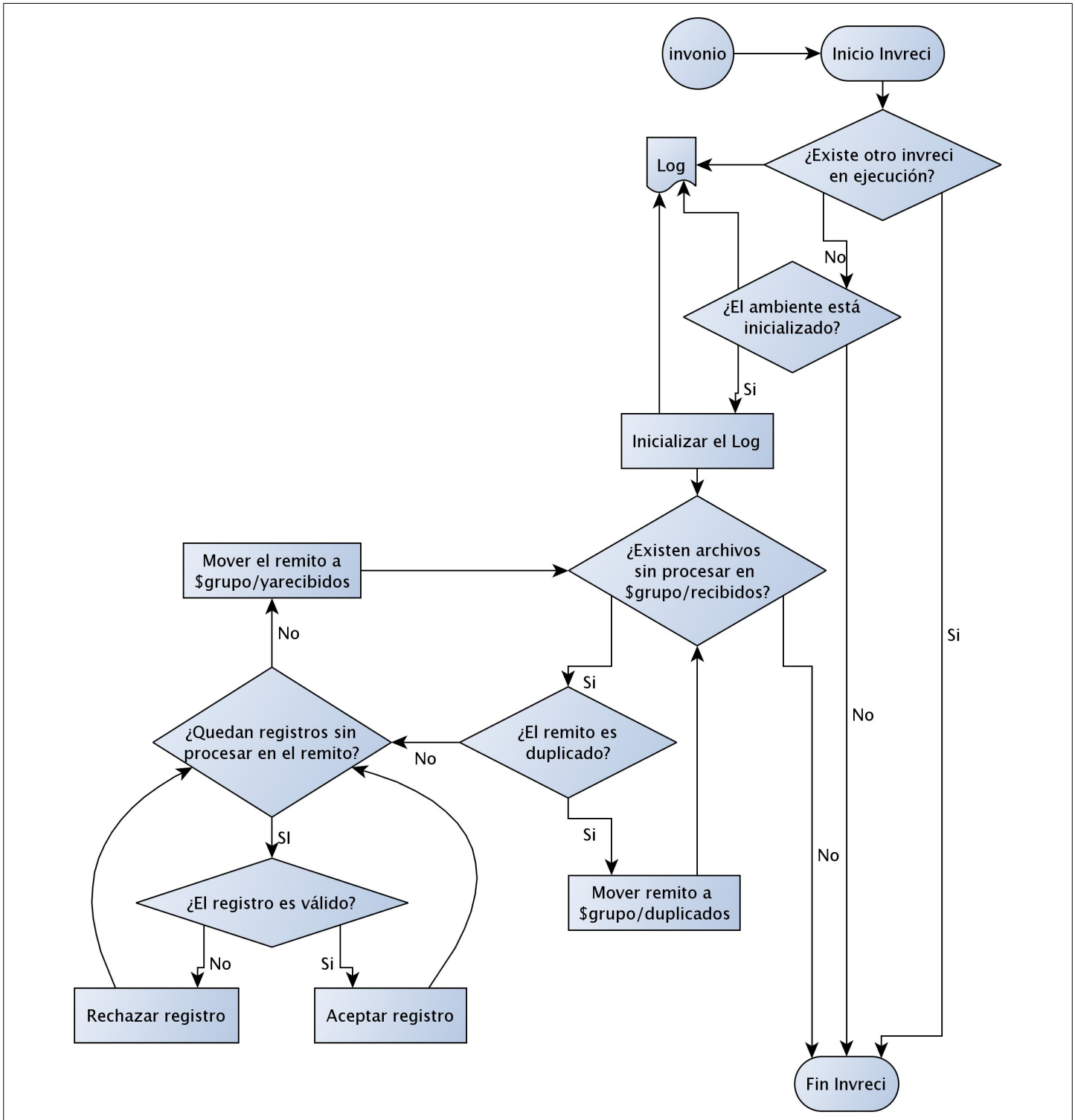

Diagrama de Proceso 2: Comando Invreci

- ***Diagrama de Proceso: Comando remioc***

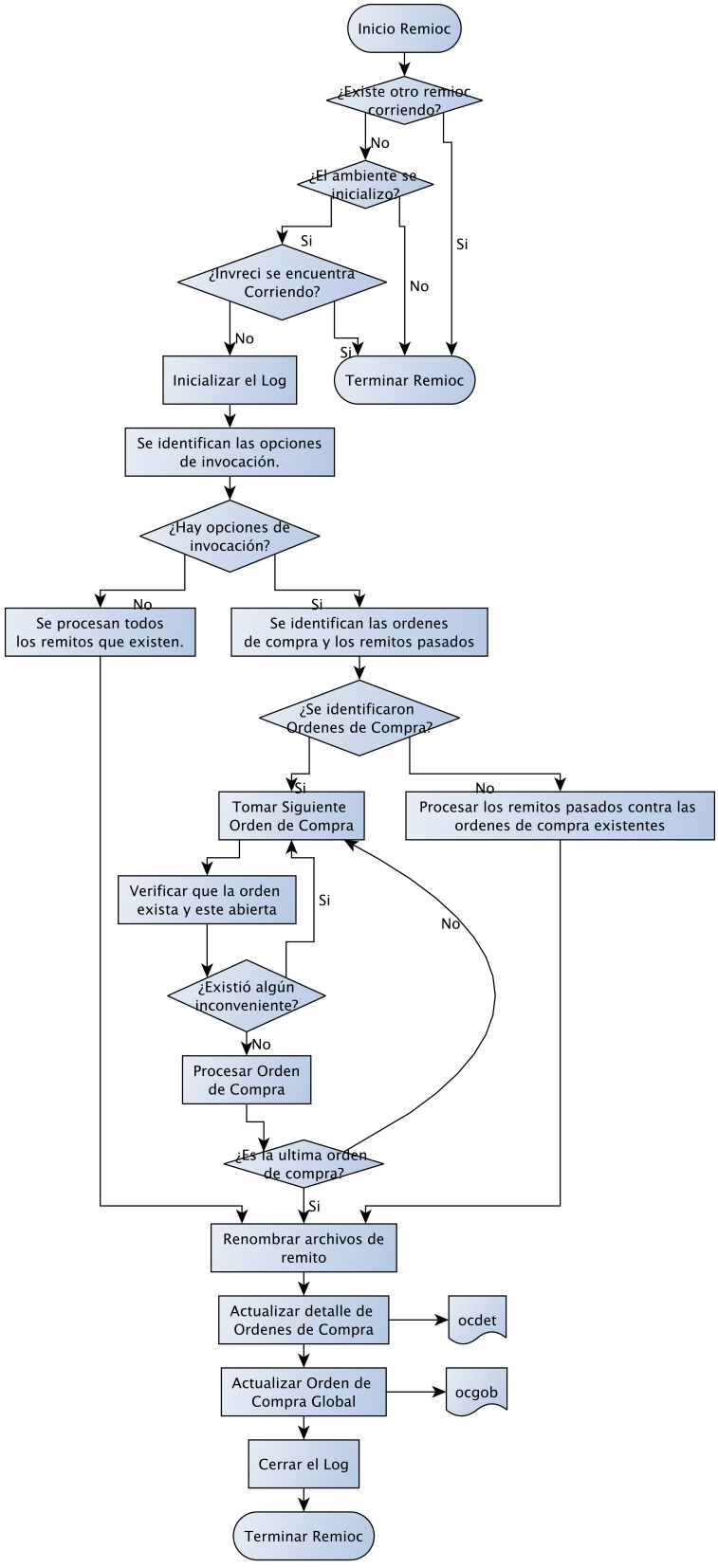

Diagrama de Proceso 3: Comando Remioc

- ***Diagrama de Proceso: Comando occtrl***

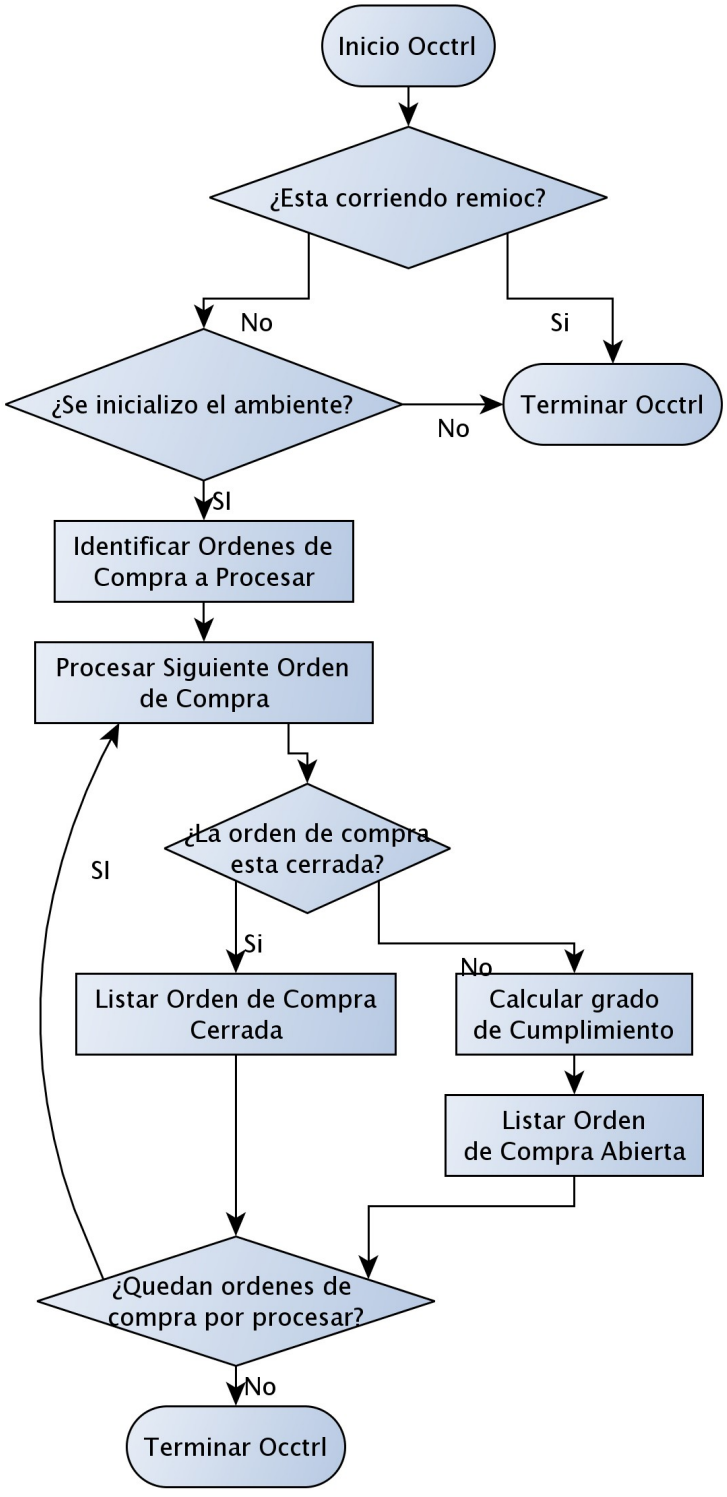

Diagrama de Proceso 4: Comando Occtrl

- ***Diagrama de Proceso: Proceso General***

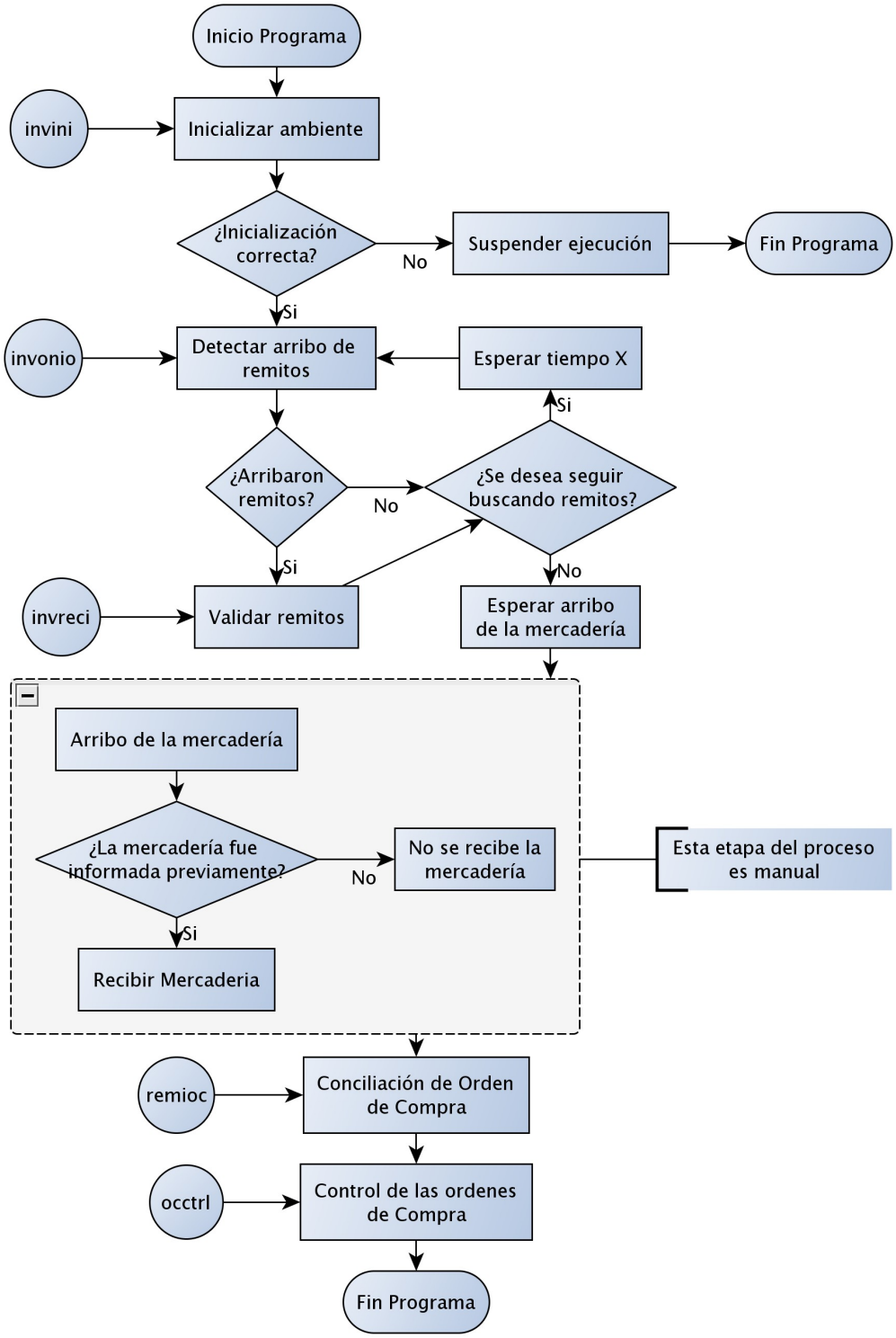

Diagrama de Proceso 5: General

Hoja de Ruta para la Corrección

Comandos Solicitados

Comandos Auxiliares

- ***Nombre del comando: bloquearProceso***

Justificación de su uso:	Este comando se utiliza para permitir ejecutar un comando sólo una vez y no permitir ejecutar el mismo comando en simultáneo, para ello el mismo genera un archivo de lock.
Archivos de Input:	Ninguno.
Archivos de Output:	Si es la primera vez que se lo ejecuta, es decir, si este comando no se encontraba ejecutándose en ese momento, genera el archivo de lock. Éste archivo conforma su nombre de la siguiente manera: ".lock_comandoqueloinvoca_corriendo" . Como puede observarse se trata de un archivo oculto.
Parámetros:	Nombre del comando que lo invoca.
Opciones:	Ninguna.

Ejemplo de invocación:	`bloquearProceso "$0"`
	`if [ $? -ne 0 ]`
	`then`
	`  echo Ya existe un proceso $0 corriendo, se termina la ejecucion.`
	`fi`
	`-Donde $0 es el comando que lo invoca.`

Listado del código:	`bloquearProceso(){` `  if [ $# -ne 1 ]` `  then` `  return 1` `  fi` `  mkdir -p "$grupo/locks"` `  local nombre=`basename "$1"` ` local ARCHIVO_LOCK="$grupo/locks/.lock_${nombre}_corriendo"` ` if [-e "$ARCHIVO_LOCK"]` ` then` ` return 2` ` fi`  ` # Creo un archivo lock oculto con el numero de pid del proceso` ` echo PID=$$ > "$ARCHIVO_LOCK"`

```
    return 0
}
```

- ***Nombre del comando: desbloquearProceso***

Justificación de su uso:	Este comando se utiliza para desbloquear un proceso bloqueado mediante el comando bloquearProceso. Para ello elimina el archivo de lock generado por éste último.
Archivos de Input:	Archivo de lock del proceso invocante (si existe).
Archivos de Output:	Ninguno.
Parámetros:	Nombre del comando que lo invoca.
Opciones:	Ninguna.

Ejemplo de invocación:	`# Elimino el archivo lock oculto`
	`desbloquearProceso "$0"`
	`-Donde $0 es el nombre del comando que lo invoca.`

Listado del código:	`desbloquearProceso(){` `  if [ $# -ne 1 ]` `  then` `  return 1` `  fi` `  local nombre=`basename "$1"` ` local ARCHIVO_LOCK="$grupo/locks/.lock_${nombre}_corriendo"`  ` rm -rf "$ARCHIVO_LOCK" > /dev/null` ` return $?` `}`

- ***Nombre del comando: estaCorriendo***

Justificación de su uso:	Éste comando se utiliza para verificar si el proceso ya se encuentra corriendo en ese momento. Se lo utiliza en conjunto con los comandos bloquearProceso y desbloquearProceso.
Archivos de Input:	Ninguno.
Archivos de Output:	Ninguno.
Parámetros:	Nombre del comando que lo invoca.
Opciones:	Ninguna.
Ejemplo de invocación:	En perl: `/bin/bash -c "estaCorriendo invreci"`; Este comando chequea si esta corriendo el comando invreci en ese momento.

Listado del código:	estaCorriendo(){
	if [$# -ne 1]
	then
	return 1
	fi
	
	local nombre=`basename "$1"`
	local ARCHIVO_LOCK="$grupo/locks/.lock_${nombre}_corriendo"
	
	if [-e "$ARCHIVO_LOCK"]
	then
	return 0
	fi
	return 2
	}

- ***Nombre del comando: getFechayHora***

Justificación de su uso:	Se lo utiliza para generalizar la obtención de la fecha y hora que es necesaria para grabar los registros durante la ejecución de los comandos.
Archivos de Input:	Ninguno.
Archivos de Output:	Ninguno.
Parámetros:	Ninguno.
Opciones:	Ninguna.

Ejemplo de invocación:	En perl:
	`my $fecha=`/bin/bash -c getFechaYHora`;`
	`print "fecha: $fecha\n";`

Listado del código:	`getFechaYHora(){`
	`  echo -n `date +"%Y/%m/%d %H:%M:%S"``
	`  return $?`
	`}`

- ***Nombre del comando: getUsuario***

Justificación de su uso:	Se utiliza para generalizar la obtención del usuario necesario para grabar los registros durante la ejecución de los comandos.
Archivos de Input:	Ninguno.
Archivos de Output:	Ninguno.
Parámetros:	Ninguno.
Opciones:	Ninguna.

Ejemplo de invocación:	En perl:
	`my $usuario=`/bin/bash -c getUsuario`;`
	`print "Usuario: $usuario\n";`

Listado del código:	`getUsuario(){`
	`  echo -n `id -un``
	`  return $?`
	`}`

Archivos Auxiliares

- ***Nombre del archivo: .lock_invonio_corriendo***

Estructura:	En su interior se guarda el PID del proceso invonio que se ejecutó.
Justificación de uso:	Es un archivo de lock que se utiliza para saber si ya se encuentra en ejecución el comando invonio en ese momento, para ello se valida la existencia de dicho archivo y al finalizar el proceso se lo elimina para permitir que se pueda correr nuevamente. En especial el archivo .lock_invonio_corriendo utiliza el PID para luego desde la función ./stopinvonio puede realizarse un SIGTERM sobre el proceso invonio que se inicializó anteriormente.

- ***Nombre del archivo: .lock_invreci_corriendo***

Estructura:	En su interior se guarda el PID del proceso invreci que se ejecutó.
Justificación de uso:	Es un archivo de lock que se utiliza para saber si ya se encuentra en ejecución el comando invreci en ese momento, para ello se valida la existencia de dicho archivo y al finalizar el proceso se lo elimina para permitir que se pueda correr nuevamente.

- ***Nombre del archivo: .lock_remioc_corriendo***

Estructura:	En su interior se guarda el PID del proceso remioc que se ejecutó.
Justificación de uso:	Es un archivo de lock que se utiliza para saber si ya se encuentra en ejecución el comando remioc en ese momento, para ello se valida la existencia de dicho archivo y al finalizar el proceso se lo elimina para permitir que se pueda correr nuevamente.

- ***Nombre del archivo: sobrante.sob***

Estructura:	**Formato de registros** **Campo** — **Descripción** **Número de orden de compra** — 6 caracteres **Código de producto a entregar** — 10 caracteres **Cantidad sobrante** — Numérico, mayor que cero **Usuario de grabación** — N caracteres. Usuario que graba el registro **Fecha y hora de grabación** — N caracteres. Fecha y hora de grabación de registro
Justificación de uso:	Se utiliza para resguardar los registros que no tienen un lugar donde registrarse para de este modo evitar perder la información correspondiente a los mismos. Cuando sobran productos, se guardan los datos en un archivo /$grupo/sobrantes/sobrante.sob/.

Apéndice A