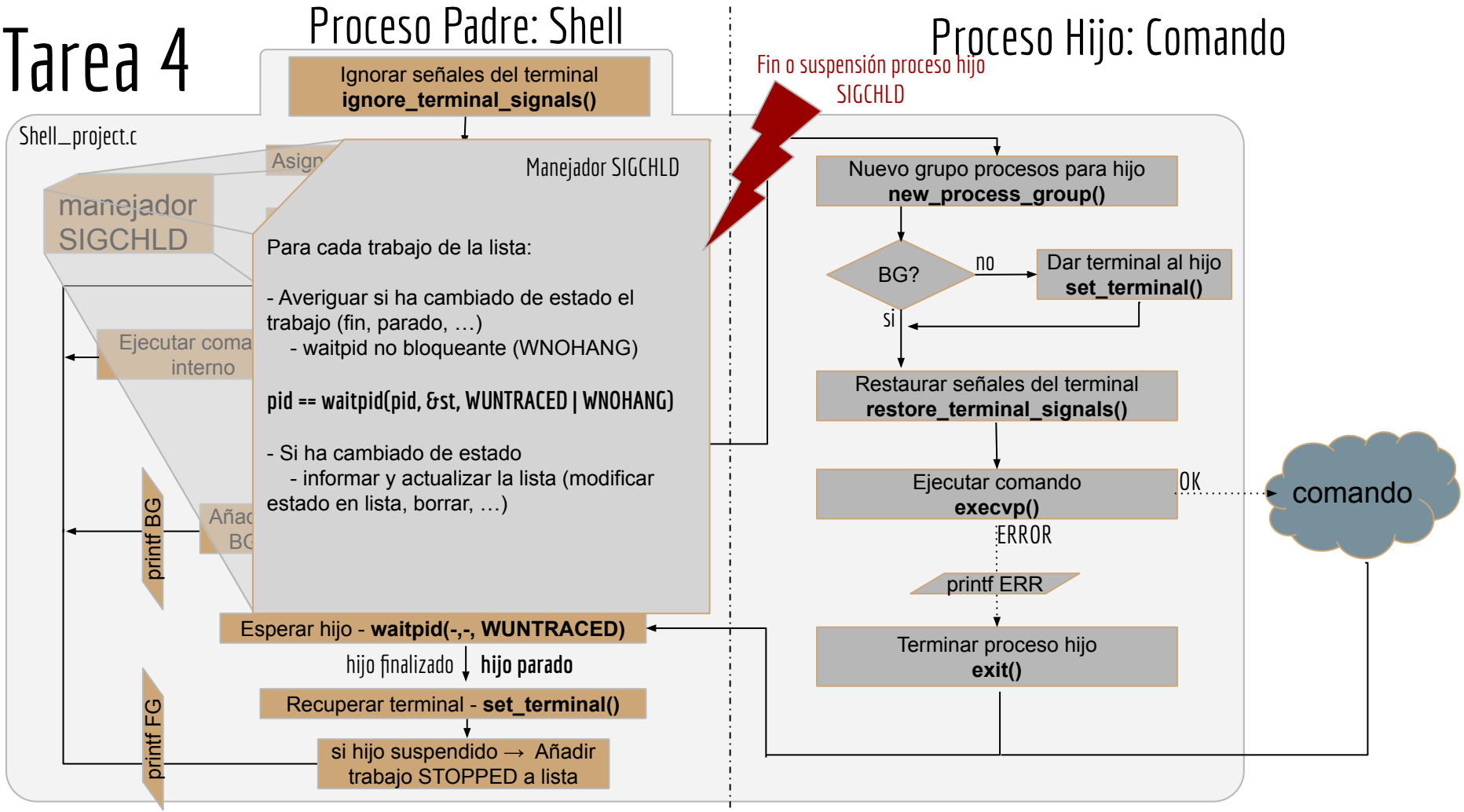
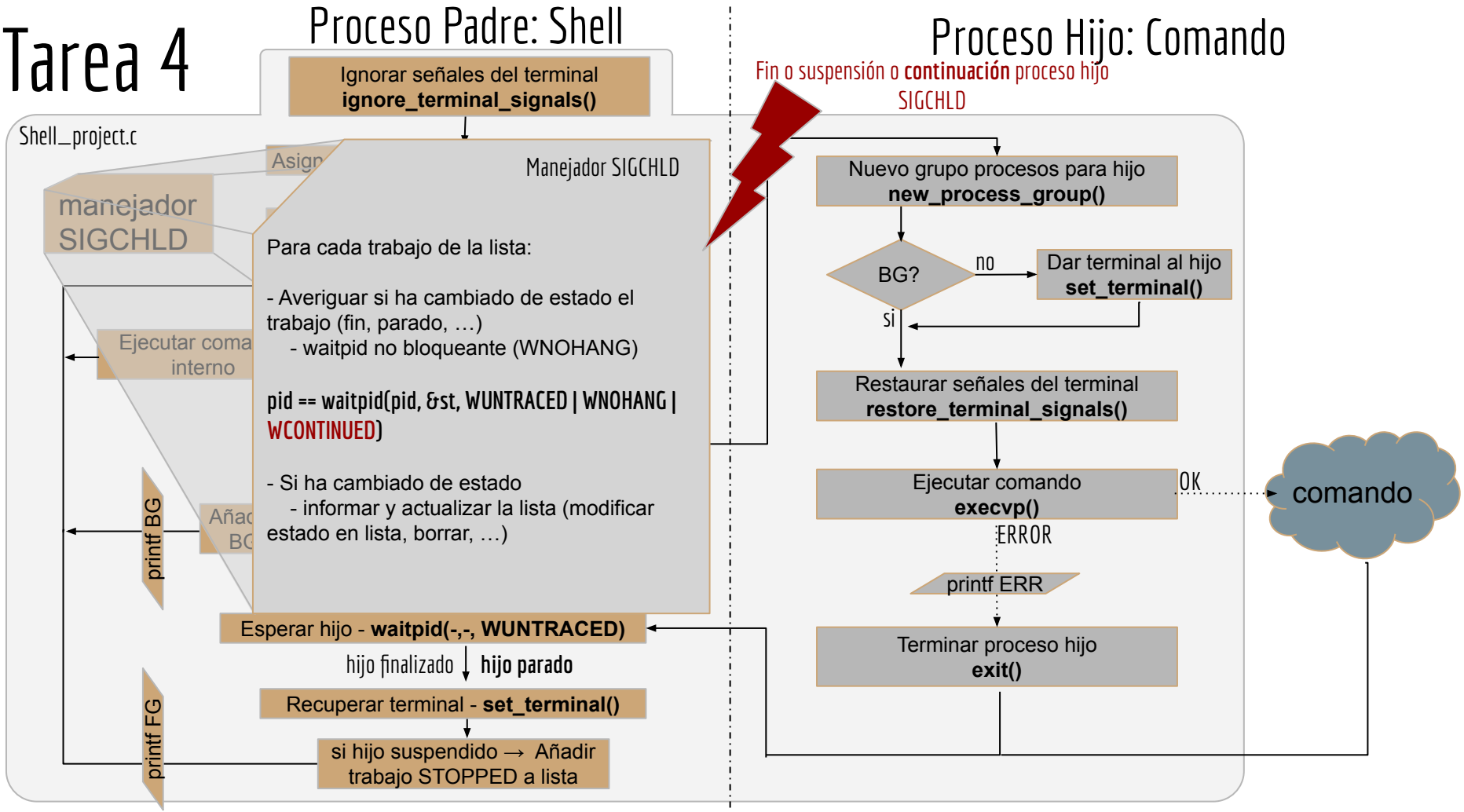
Proyecto
SHELL básico

Tarea 4

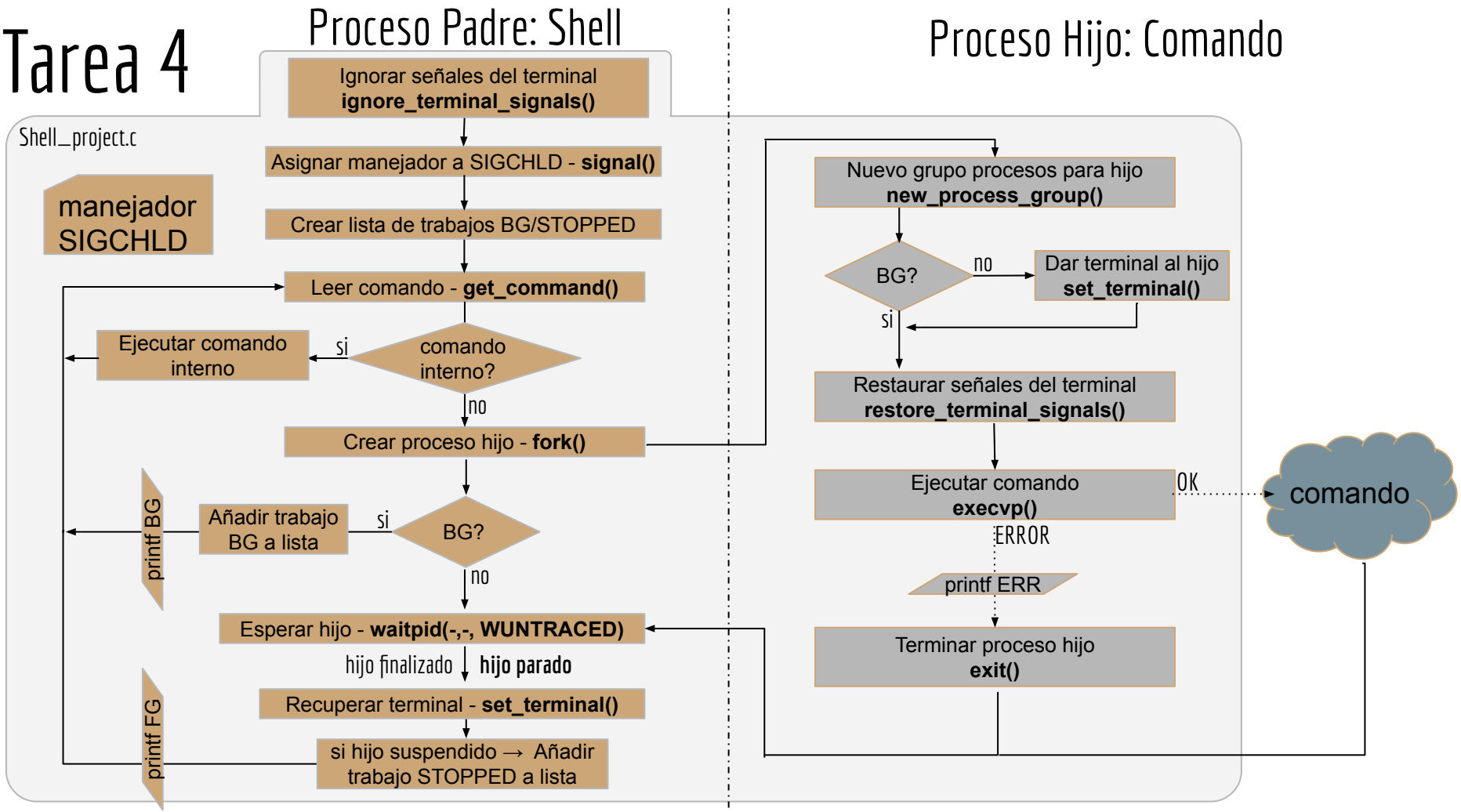
Tarea 4


Proceso Padre: Shell
Proceso Hijo: Comando
Ignorar señales del terminal
ignore_terminal_signals()
Shell_project.c
Fin o suspensión proceso hijo
SIGCHLD
Manejador SIGCHLD
manejador SIGCHLD
Para cada trabajo de la lista:
- Averiguar si ha cambiado de estado el trabajo (fin, parado, …)
- waitpid no bloqueante (WNOHANG)
pid == waitpid(pid, &st, WUNTRACED | WNOHANG)
- Si ha cambiado de estado
- informar y actualizar la lista (modificar estado en lista, borrar, …)
Nuevo grupo procesos para hijo
new_process_group()
BG?
no
Dar terminal al hijo
set_terminal()
si
Restaurar señales del terminal
restore_terminal_signals()
Ejecutar comando
execvp()
OK
comando
ERROR
printf ERR
Terminar proceso hijo
exit()
printf BG
printf FG
Esperar hijo - waitpid(-,-, WUNTRACED)
hijo finalizado
hijo parado
Recuperar terminal - set_terminal()
si hijo suspendido → Añadir trabajo STOPPED a lista

Tarea 4


Proceso Padre: Shell
Proceso Hijo: Comando
Ignorar señales del terminal
ignore_terminal_signals()
Shell_project.c
Fin o suspensión o continuación proceso hijo
SIGCHLD
Manejador SIGCHLD
manejador SIGCHLD
Para cada trabajo de la lista:
- Averiguar si ha cambiado de estado el trabajo (fin, parado, …)
- waitpid no bloqueante (WNOHANG)
pid == waitpid(pid, &st, WUNTRACED | WNOHANG | WCONTINUED)
- Si ha cambiado de estado
- informar y actualizar la lista (modificar estado en lista, borrar, …)
Ejecutar coma interno
printf BG
printf FG
Esperar hijo - waitpid(-,-, WUNTRACED)
hijo finalizado
hijo parado
Recuperar terminal - set_terminal()
si hijo suspendido → Añadir trabajo STOPPED a lista
Nuevo grupo procesos para hijo
new_process_group()
BG?
no
Dar terminal al hijo
set_terminal()
si
Restaurar señales del terminal
restore_terminal_signals()
Ejecutar comando
execvp()
OK
comando
ERROR
printf ERR
Terminar proceso hijo
exit()

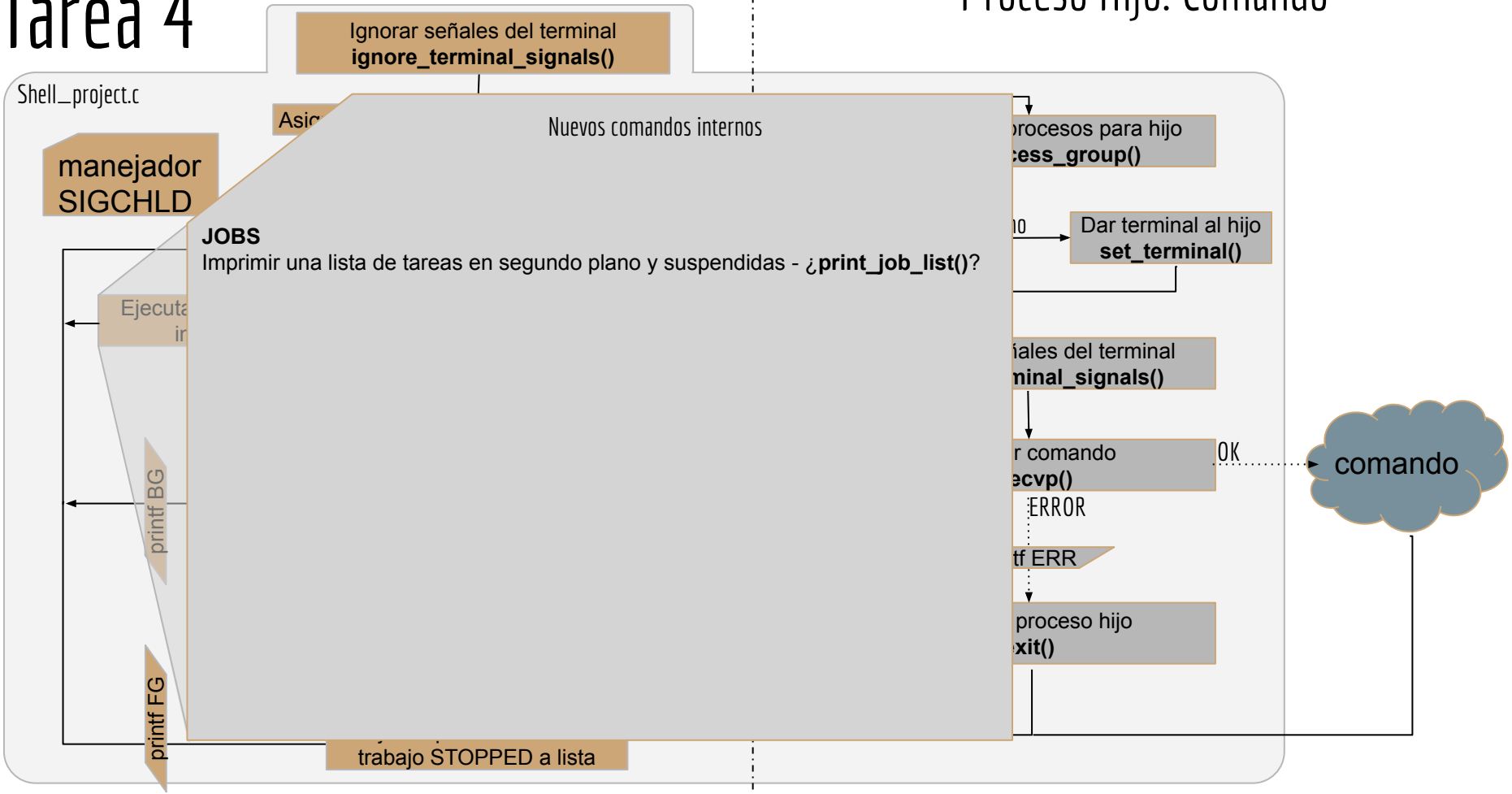
Tarea 4


Proceso Padre: Shell
Proceso Hijo: Comando
Shell_project.c
manejador SIGCHLD
Ignorar señales del terminal
ignore_terminal_signals()
Asignar manejador a SIGCHLD - signal()
Crear lista de trabajos BG/STOPPED
Leer comando - get_command()
comando interno?
si
Ejecutar comando interno
no
Crear proceso hijo - fork()
BG?
si
Añadir trabajo BG a lista
printf BG
no
Esperar hijo - waitpid(-,-, WUNTRACED)
hijo finalizado
hijo parado
Recuperar terminal - set_terminal()
si hijo suspendido → Añadir trabajo STOPPED a lista
printf FG
Nuevo grupo procesos para hijo
new_process_group()
BG?
no
Dar terminal al hijo
set_terminal()
si
Restaurar señales del terminal
restore_terminal_signals()
Ejecutar comando
execvp()
OK
ERROR
printf ERR
Terminar proceso hijo
exit()
comando

Tarea 4

Proceso Padre: Shell

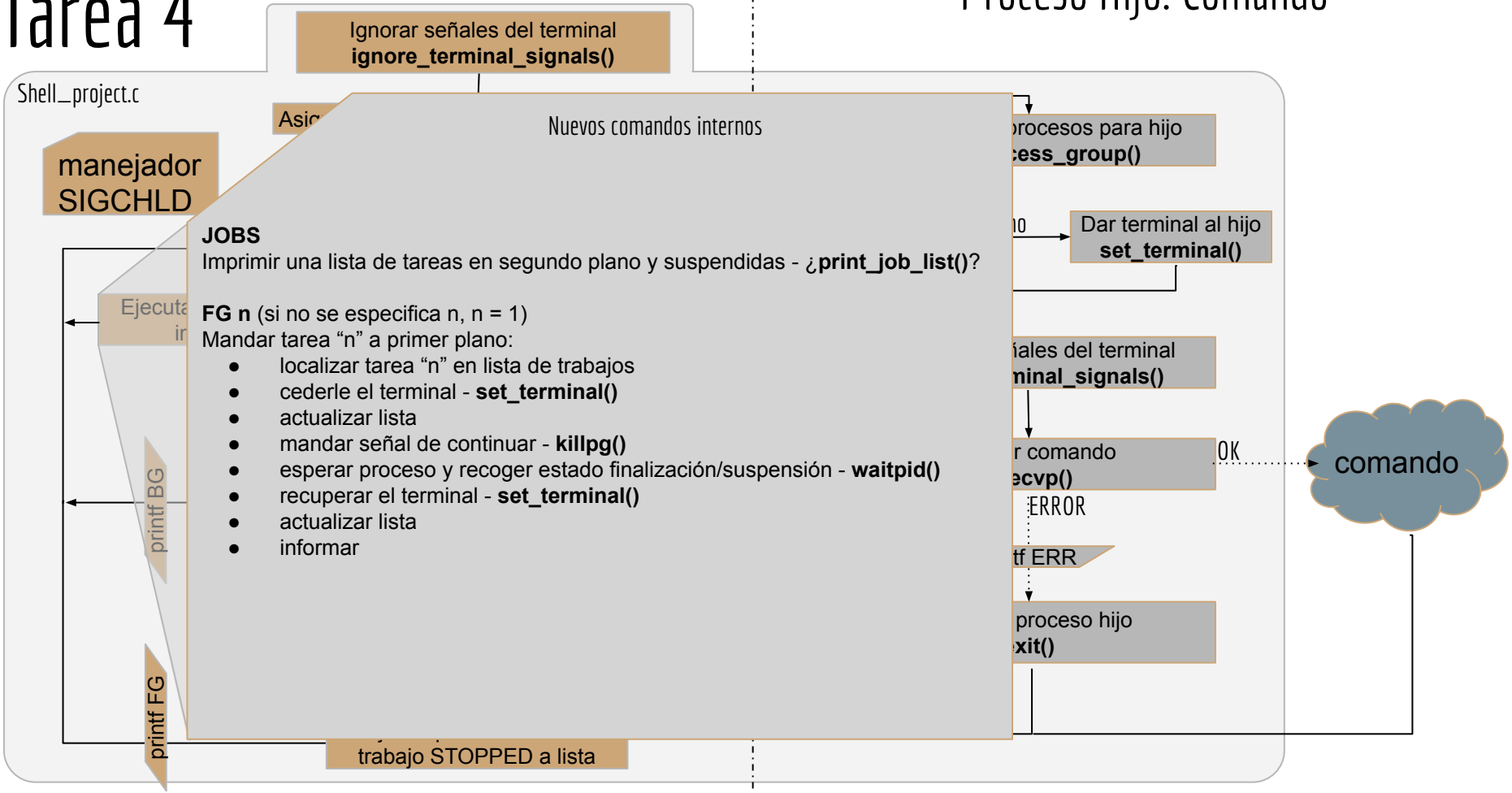
Proceso Hijo: Comando



Tarea 4

Proceso Padre: Shell

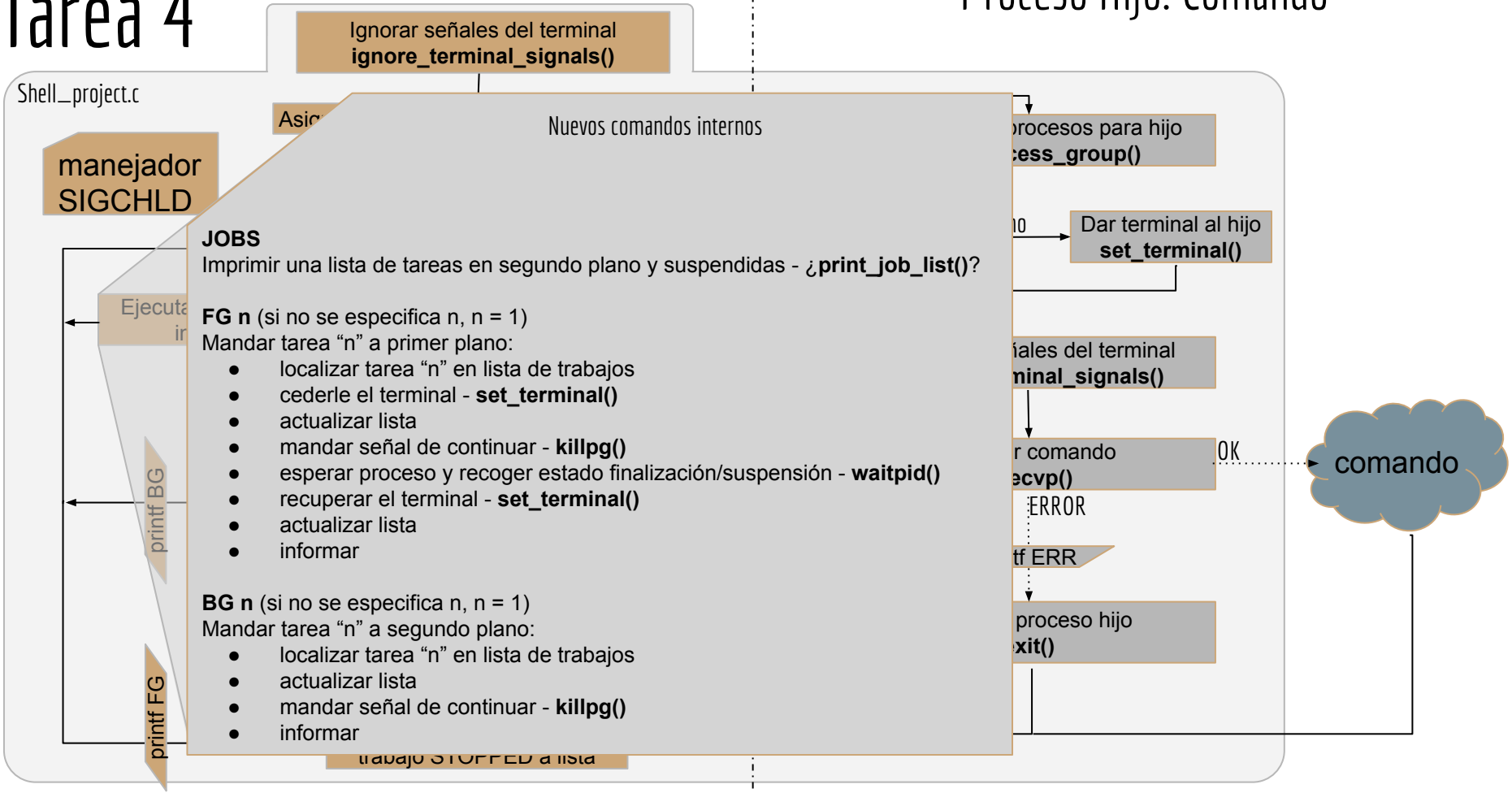
Proceso Hijo: Comando



Tarea 4

Proceso Padre: Shell

Proceso Hijo: Comando



Nuevos comandos internos

JOBS
Imprimir una lista de tareas en segundo plano y suspendidas - ¿**print_job_list()**?

FG n (si no se especifica n, n = 1)
Mandar tarea “n” a primer plano:

- localizar tarea “n” en lista de trabajos
- cederle el terminal - **set_terminal()**
- actualizar lista
- mandar señal de continuar - **killpg()**
- esperar proceso y recoger estado finalización/suspensión - **waitpid()**
- recuperar el terminal - **set_terminal()**
- actualizar lista
- informar

BG n (si no se especifica n, n = 1)
Mandar tarea “n” a segundo plano:

- localizar tarea “n” en lista de trabajos
- actualizar lista
- mandar señal de continuar - **killpg()**
- informar