Checklist de Validación: Trabajo Práctico de Sistemas Operativos (Shell)

Enfoque del Proyecto: Shell con énfasis en Seguridad (Sandboxing y Auditoría). **Estado General:** Código Avanzado / Documentación Completa.

1. Arquitectura y Definición del Enfoque

Requisito del TP	Estado en tu Código (`flsh_shell.c`)	Acción Requerida / Observación
Tema/Enfoque Único (No genérico)	✅ **Cubierto.** Implementas "Sandboxing" en `$HOME` y logs de seguridad.	Asegurate de defender este punto en la presentación oral.
Prompt Personalizado	✅ **Cubierto.** Muestra `[Úsame: /ruta]>`.	-
Lenguaje C puro	✅ **Cubierto.** Uso correcto de librerías estándar y syscalls.	-
Sin librerías prohibidas (`system`, `popen`)	✅ **Cubierto.** No se detectaron funciones prohibidas.	-
Manejo de Errores (`errno`)	✅ **Cubierto.** Función `reportar_error_sistema` usa `strerror(errno)`.	Excelente práctica implementada.

2. Comandos Internos (Built-ins)

Se verificó que no se invoquen binarios externos.

Comando	Lógica Implementada	Estado	Notas del Experto
`ls`	Uso de `opendir`, `readdir`. Filtra ocultos.	✅ Listo	Verifica si el profesor pide explícitamente flag `-a` (ocultos). Actualmente los ocultas por defecto.
`cd`	Uso de `chdir` y `putenv` para `$PWD`.	✅ Listo	Valida el caso `cd` sin argumentos (debería ir a HOME). Tu código actual lo hace.
`cp`	Lectura/Escritura con buffer (syscalls `read`/`write`).	✅ Listo	Tienes un *Safety Interlock* (pregunta si sobrescribe). ¡Gran "plus" para el enfoque de seguridad!
`rm`	Uso de syscall `unlink`.	✅ Listo	También tiene confirmación de seguridad. Correcto.
`mkdir`	Uso de syscall `mkdir` con permisos 0755.	✅ Listo	Maneja correctamente el error si ya existe.
`echo`	Imprime argumentos y salta línea.	✅ Listo	Soporta redirección gracias a la lógica del `main`.
`cat`	Lectura/Escritura directa a `STDOUT`.	✅ Listo	Muy eficiente.
`pwd`	Uso de `getcwd`.	✅ Listo	-

`exit`	Termina el loop y loguea salida.	✅ Listo	-
`grep` (Opcional +2pts)	Uso de `strstr` y `fgets`. Cuenta coincidencias.	✅ Listo	Implementación nativa correcta.

3. Ejecución de Procesos Externos

Requisito	Implementación	Estado	Observación
Fork/Exec/Wait	Ciclo estándar UNIX implementado.	✅ Listo	Manejas correctamente el proceso hijo y el padre espera.
Manejo de Errores	Detecta fallo en `execvp` y hace `exit(127)`.	✅ Listo	Esto evita procesos zombies o shells colgadas.
Sandboxing Externo	Validas rutas de argumentos en binarios externos.	✅ Listo	Coherente con tu enfoque de seguridad.

4. Gestión de I/O y Redirección

Requisito	Implementación	Estado	Observación
Redirección `>`	Manipulación de File Descriptors (`dup`, `dup2`).	✅ Listo	Funciona para internos y externos.
Limpieza de Buffer	Uso de `fflush(stdout)`.	✅ Listo	Vital para que el prompt y `printf` no se mezclen.

Restauración	Recupera `STDOUT` original tras el comando.	✅ Listo	Correctamente implementado en el `main`.

5. Sistema de Logs y Auditoría (Tu Fuerte)

Requisito	Implementación	Estado	Observación
Archivos Separados	`shell.log` (Info) y `sistema_error.log` (Error).	✅ Listo	Cumple estrictamente el pedido.
Datos del Log	Timestamp, User, CMD, IP origen.	✅ Listo	Agregaste detección de IP (SSH vs Local), excelente valor agregado.
Persistencia	Fallback a `./logs` si `/var/log` falla.	✅ Listo	Solución robusta para entornos sin root.