Aviso: este es un workaround no oficial y específico para cuando OpenAI Codex CLI en macOS Mojave sobre Inte, falla en el proceso codex-code-mode-host. La solución conserva una copia del binario original, pero una actualización de Codex podría reemplazar los archivos modificados.

El problema

Codex CLI arrancaba correctamente, pero cualquier intento de ejecutar una herramienta, incluso un sencillo **pwd**, terminaba con este mensaje:

```
code-mode host closed its stdout
```

El problema persistía después de reiniciar Codex, crear una conversación nueva y reiniciar completamente el Mac. Por tanto, no estaba relacionado con el proyecto, el shell, los permisos del directorio ni el sandbox.

El entorno en el que se reprodujo el fallo era:

- macOS Mojave 10.14.6.
- Arquitectura Intel x86_64.
- OpenAI Codex CLI 0.151.0.
- Instalación standalone dentro de ~/.codex/.

Cómo encontramos la causa

El binario responsable existía, tenía la arquitectura correcta y podía mostrar su ayuda:

```
HOST_BIN="$(command -v codex-code-mode-host)"

file -L "$HOST_BIN"
"$HOST_BIN" --help
```

Lo pudimos ver en la salida:

```
/Users/nipegun/.local/bin/codex-code-mode-host: POSIX shell script text executable, ASCII text

Usage: codex-code-mode-host.real [OPTIONS]

Options:
  --listen
    Transport endpoint: `stdio`, `stdio://`, or `grpc://IP:PORT` [default:stdio]
  --otel-trace-listen
    Optional WebSocket endpoint that streams only raw OTLP trace batches
  --otel-trace-exporter
    Optional OTLP/HTTP JSON trace exporter endpoint, analogous to `otel.trace_exporter` in app-server configuration
  -h, --help
    Print help
```

Sin embargo, macOS había generado un informe de bloqueo en **~/Library/Logs/DiagnosticReports**. Las líneas decisivas se obtuvieron con:

```
CRASH_FILE="$(find "$HOME/Library/Logs/DiagnosticReports" \ -maxdepth 1 \ -type f \ -name 'codex-code-mode-host*' \
-print | sort | tail -1)"
grep -E '^(Path:|Date/Time:|Termination Reason:|Dyld Error Message:| Symbol not found:| Referenced from:| Expected
in:)' "$CRASH_FILE"
```

El informe mostró:

```
Termination Reason: DYLD, [0x4] Symbol missing

  Dyld Error Message:
    Symbol not found: _aligned_alloc
    Referenced from: codex-code-mode-host
    Expected in: /usr/lib/libSystem.B.dylib
```

El host incorpora V8 para ejecutar Code Mode. El programa podía procesar **–help**, pero al crear el entorno de V8 intentaba utilizar **aligned_alloc**. Ese símbolo no está exportado por **libSystem** en Mojave, por lo que **dyld** abortaba el proceso.

Por qué no bastaba con desactivar Code Mode

Probamos a iniciar Codex con:

```
codex --disable code_mode_host
```

Esto evitaba el crash, pero no permitía ejecutar herramientas. La versión utilizada falla de forma cerrada cuando el host está desactivado y no cambia automáticamente al ejecutor clásico:

```
Code Mode is unavailable because code-mode host is disabled.
Code mode will fail closed.
```

Por tanto, necesitábamos hacer funcionar el host, no simplemente desactivarlo.

La solución aplicada

Creamos una pequeña biblioteca de compatibilidad que implementa **aligned_alloc** mediante **posix_memalign**, disponible en Mojave. La memoria obtenida sigue siendo compatible con **free()**.

1. Compilar la biblioteca de compatibilidad

Este paso necesita las herramientas de línea de comandos de Xcode:

```
COMPAT_DIR="$HOME/.codex/mojave-compat"
mkdir -p "$COMPAT_DIR"

/usr/bin/xcrun clang -dynamiclib -x c -arch x86_64 -mmacosx-version-min=10.14 -o "$COMPAT_DIR/libcodex-mojave-
compat.dylib" - <<'EOF'
#include <errno.h>
#include <stddef.h>
#include <stdlib.h>

void *aligned_alloc(size_t alignment, size_t size)
{
  void *pointer = NULL;

  if (alignment == 0 ||
    (alignment & (alignment - 1)) != 0 ||
    size % alignment != 0) {
      errno = EINVAL;
      return NULL;
  }

  int result = posix_memalign(&pointer, alignment, size);
  if (result != 0) {
```

```
    errno = result;
    return NULL;
  }

  return pointer;
}
EOF
```

2. Preparar una copia del host

Codex no buscaba el host únicamente mediante **PATH**, sino que ejecutaba directamente el binario incluido junto a la instalación. Por eso fue necesario envolver ese archivo exacto.

```
REAL_HOST="$(readlink "$HOME/.local/bin/codex-code-mode-host")"
COMPAT_DIR="$HOME/.codex/mojave-compat"

cp "$REAL_HOST" "$COMPAT_DIR/codex-code-mode-host.real"
chmod u+w "$COMPAT_DIR/codex-code-mode-host.real"

/usr/bin/codesign --remove-signature "$COMPAT_DIR/codex-code-mode-host.real" 2>/dev/null
```

La firma se elimina solamente de la copia utilizada por el workaround. El binario original permanece intacto como respaldo.

3. Guardar el original e instalar el wrapper

```
mv "$REAL_HOST" "${REAL_HOST}.original-mojave"

cat > "$REAL_HOST" <<'EOF'
#!/bin/sh

COMPAT_DIR="$HOME/.codex/mojave-compat"

export DYLD_INSERT_LIBRARIES="$COMPAT_DIR/libcodex-mojave-compat.dylib"
export DYLD_FORCE_FLAT_NAMESPACE=1

exec "$COMPAT_DIR/codex-code-mode-host.real" "$@"
EOF

chmod 755 "$REAL_HOST"
```

El wrapper carga la biblioteca de compatibilidad y ejecuta la copia del host. Gracias a **DYLD_FORCE_FLAT_NAMESPACE**, V8 puede resolver **aligned_alloc** desde nuestra biblioteca en lugar de buscarlo exclusivamente en **libSystem**.

4. Verificar el resultado

```
cd /ruta/al/proyecto
codex
```

Dentro de Codex solicitamos:

```
Ejecuta pwd
```

Esta vez la herramienta se ejecutó correctamente:

```
Ran pwd
└ /Users/usuario/Desktop/Code
```

Cómo deshacer el cambio

El binario original queda guardado con el sufijo **.original-mojave**. Para restaurarlo sin borrar el wrapper:

```
REAL_HOST="$(readlink "$HOME/.local/bin/codex-code-mode-host")"

mv "$REAL_HOST" "${REAL_HOST}.wrapper-mojave"
mv "${REAL_HOST}.original-mojave" "$REAL_HOST"
```

Después de comprobar que Codex vuelve a arrancar, el directorio **~/.codex/mojave-compat** y el wrapper guardado pueden eliminarse manualmente si ya no son necesarios.

Limitaciones

- Es un workaround no oficial para Codex CLI 0.151.0.
- Una actualización puede reemplazar el wrapper.
- No debe reutilizarse una copia antigua del host con una versión nueva de Codex sin verificar que sus protocolos siguen siendo compatibles.
- Si el informe de bloqueo menciona un símbolo diferente, esta solución podría no ser suficiente.
- La solución definitiva sería distribuir un host compilado para Mojave o implementar oficialmente el fallback mediante **posix_memalign**.