Cómo logré escribir en un controller pak de N64 usando libdragon

Enlace al controller pak.

Este post documenta el camino para programar de forma nativa en una Nintendo 64, superando errores de arquitectura en Mac M1/M2 y problemas físicos de detección del hardware.

1□□ El Problema: Docker y la arquitectura ARM

En Mac modernos (Apple Silicon), la imagen oficial de Libdragon suele dar errores de plataforma. Muchos intentan:

Bash

```
libdragon init
```

Y reciben el error: no matching manifest for linux/arm64/v8.

La Solución (Comandos de limpieza y forzado):

Para solucionar esto, hay que limpiar el entorno y forzar la plataforma compatible:

Bash

```
# Limpiar contenedores e imágenes fallidas
docker ps -a | grep libdragon | awk '{print $1}' | xargs docker rm -f
docker rmi -f ghcr.io/dragonminded/libdragon:latest
docker system prune -f

# Forzar plataforma AMD64 para que funcione en Mac ARM
```

```
export DOCKER_DEFAULT_PLATFORM=linux/amd64
libdragon init
```

2□□ El Desafío del Hardware: ¿Rumble o Memoria?

Un problema crítico fue que el **Controller Pak** (Memoria) era detectado erróneamente como un **Rumble Pak** (Vibración). Esto bloquea las funciones normales de guardado porque el sistema «cree» que le está hablando a un motor y no a un chip de memoria.

La solución técnica:

Ignorar la detección automática y forzar el acceso a bajo nivel con **Joybus**, asegurando que los datos en la RAM estén perfectamente alineados.

□ 3□□ El Makefile Maestro

Este archivo asegura que el compilador MIPS de la N64 se utilice correctamente, evitando que el Mac intente usar su propio compilador.

Makefile

```
# Rutas internas del contenedor/toolchain
N64_INST = /n64_toolchain
LIBDRAGON_PATH = /libdragon

CC = $(N64_INST)/bin/mips64-elf-gcc
AS = $(N64_INST)/bin/mips64-elf-as
LD = $(N64_INST)/bin/mips64-elf-ld
```

```
OBJCOPY = $(N64_INST)/bin/mips64-elf-objcopy

include $(N64_INST)/include/n64.mk

# Importante: -falign-functions=32 para estabilidad en hardware real
CFLAGS += -I$(LIBDRAGON_PATH)/include -falign-functions=32

PROG_NAME = hello
OBJS = build/main.o

all: $(PROG_NAME).z64

$(PROG_NAME).elf: $(OBJS)

$(PROG_NAME).z64: N64_ROM_TITLE = "Controller Pak Test"

build/main.o: src/main.c
	@mkdir -p build
	$(CC) $(CFLAGS) -c -o $@ $<

clean:
	rm -rf build *.z64 *.elf
.PHONY: clean
```

🎮 4️⃣ El Código C (Fuerza Bruta y Alineación)

Ubicación: src/main.c. Este código es el «tanque» que logramos hacer funcionar.

C

```
#include <stdio.h>
#include <string.h>
#include <libdragon.h>

int main(void) {
```

```
    timer_init();
      display_init(RESOLUTION_320x240, DEPTH_16_BPP, 2,
GAMMA_NONE, ANTIALIAS_RESAMPLE);
    joypad_init();

    char status[64] = "Listo para Guardar/Leer";
    char lectura[32] = "---";
    // CLAVE: El buffer DEBE estar alineado a 16 bytes para
evitar corromper el Joybus
    uint8_t buffer_seguro[32] __attribute__((aligned(16)));

    while (1) {
        joypad_poll();
                              joypad_buttons_t   btn   =
joypad_get_buttons_pressed(JOYPAD_PORT_1);
                        joypad_accessory_type_t  acc   =
joypad_get_accessory_type(JOYPAD_PORT_1);

        surface_t *disp = display_get();
        graphics_fill_screen(disp, 0);

         graphics_draw_text(disp, 20, 20, "== N64 CONTROLLER
PAK DEBUG ==");

        // Si se detecta mal (como Rumble), el programa sigue
funcionando igual
        if (acc == ACCESSORY_RUMBLEPAK) {
            graphics_draw_text(disp, 40, 50, "AVISO: Detectado
como Rumble (Incorrecto)");
        }

        // ESCRITURA FORZADA (Boton A)
        if (btn.a) {
            memset(buffer_seguro, 0, 32);
            memcpy(buffer_seguro, "HOLA_N64_EXITO", 14);
            // Usamos la dirección 0x0100 (Bloque seguro fuera
del Index)
             int res = joybus_accessory_write(JOYPAD_PORT_1,
0x0100, buffer_seguro);
             if(res == 0) strcpy(status, "Status: ESCRITURA
OK");
```

```
            else sprintf(status, "Status: ERROR ESC. %d",
res);
        }

        // LECTURA FORZADA (Boton B)
        if (btn.b) {
            memset(buffer_seguro, 0, 32);
             int res = joybus_accessory_read(JOYPAD_PORT_1,
0x0100, buffer_seguro);
            if(res == 0) {
                memcpy(lectura, buffer_seguro, 31);
                strcpy(status, "Status: LECTURA OK");
            }
        }

        graphics_draw_text(disp, 20, 100, status);
        char data_info[64];
        sprintf(data_info, "Dato leido: [%s]", lectura);
        graphics_draw_text(disp, 20, 120, data_info);

        display_show(disp);
    }
}
```

🚀 5️⃣ Resumen de Pasos para Compilar

Una vez que tienes los archivos creados:

1. **Limpiar:** libdragon make clean
2. **Compilar:** libdragon make
3. **Resultado:** Obtendrás el archivo hello.z64 listo para Flashcarts (EverDrive/ED64).

Conclusiones para el post:

- **Física:** Limpia los pines del Controller Pak; una mala lectura de ID (Rumble en lugar de Memoria) es casi siempre suciedad.
- **Software:** La alineación de memoria (__attribute__((aligned(16)))) es innegociable en N64 para periféricos.
- **Docker:** El comando export DOCKER_DEFAULT_PLATFORM=linux/amd64 es tu mejor amigo en Mac Silicon.