
Sistemas Operativos I

*“A computer terminal is not an old lump of a television
with a typewriter keyboard sitting in front of it.
It is an interface connecting body and spirit with the universe
and enabling bits of it to be moved around.”*

Douglas Adams

2022

Rafael Ignacio Zurita <rafa@fi.uncoma.edu.ar>

Advertencia: Estos slides traen ejemplos.

No copiar (ctrl+c) y pegar en un shell o terminal los comandos aquí presentes.

Algunos no funcionarán, porque al copiar y pegar también van caracteres “ocultos” (no visibles pero que están en el pdf) que luego interfieren en el shell.

Sucedio en vivo :)

Conviene “escribirlos” manualmente al trabajar.

Sistemas Operativos I - Entrada / Salida

Contenido:

- Repaso de E/S arquitectura
- Diferentes dispositivos
- Controladores de dispositivos (drivers)
 - Posibilidad 1: una interfaz para cada tipo
 - Posibilidad 2: una abstracción suficiente
- Ejemplo Xinu y Linux

Sistemas Operativos I - Entrada / Salida

Dispositivos de E/S diferentes:

- Terminales e interfaces serie (ej. UART)
- Teclados
- Mouse
- Camara de video
- Disco rígido
- Placa gráfica
- Dispositivo ethernet
- Cintas de backup

Sistemas Operativos I - Entrada / Salida

Dispositivos de E/S diferentes:

- Terminales e interfaces serie (ej. UART)
- Teclados
- Mouse
- Camara de video
- Disco rígido
- Placa gráfica
- Dispositivo ethernet
- Cintas de backup

Sistemas Operativos I - Entrada / Salida

Dispositivos de E/S diferentes:

- Terminales e interfaces serie (ej. UART)
- Teclados
- Mouse
- Camara de video
- Disco rígido
- Placa gráfica
- Dispositivo ethernet
- Cintas de backup

Consultas: ¿Orientado a bloques o a bytes? ¿De entrada o salida? ¿Soporta acceso aleatorio? ¿Puede haber mas de un dispositivo físico del mismo tipo?

Sistemas Operativos I - Entrada / Salida

Tasa de transferencia de datos según dispositivo de E/S

PCI Express Version	Bandwidth *	Bandwidth ‡	Encoding	Year Released
1.0	250MB/s	(2 GB/s)	8b/10b	2002
2.0	500MB/s	(4 GB/s)	8b/10b	2007
3.0	1000MB/s	(~ 8 GB/s)	128b/130b	2010

* Per lane, in each direction ‡ Per 8 lanes, in each direction

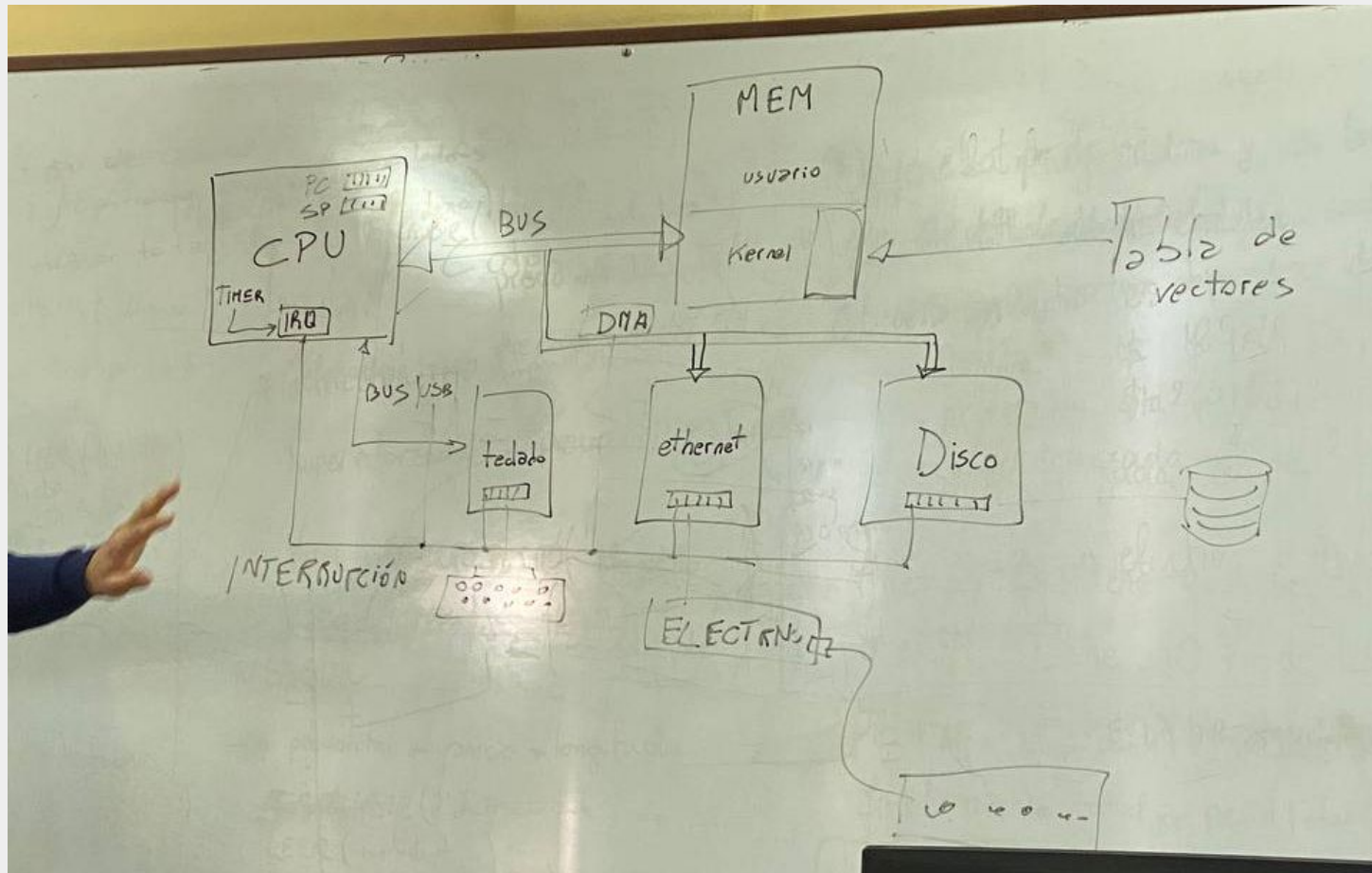
SATA revision	Bandwidth (Coded)	Bandwidth (Actual)	Year
1.0	1.5 Gbit/s	1.2 Gbit/s (150MB/s) *	2003
2.0	3 Gbit/s	2.4 Gbit/s (300MB/s) *	2005
3.0	6 Gbit/s	4.8 Gbit/s (600MB/s) *	2009

* 8b/10b encoding

Device	Data rate
Keyboard	10 bytes/sec
Mouse	100 bytes/sec
56K modem	7 KB/sec
Scanner at 300 dpi	1 MB/sec
Digital camcorder	3.5 MB/sec
4x Blu-ray disc	18 MB/sec
802.11n Wireless	37.5 MB/sec
USB 2.0	60 MB/sec
FireWire 800	100 MB/sec
Gigabit Ethernet	125 MB/sec
SATA 3 disk drive	600 MB/sec
USB 3.0	625 MB/sec
SCSI Ultra 5 bus	640 MB/sec
Single-lane PCIe 3.0 bus	985 MB/sec
Thunderbolt 2 bus	2.5 GB/sec
SONET OC-768 network	5 GB/sec

Sistemas Operativos I - Entrada / Salida

Visión general en bloques de una computadora



Sistemas Operativos I - Entrada / Salida

Mecanismos de programación de E/S

- programada (polling/consulta)
- interrupciones
- dma

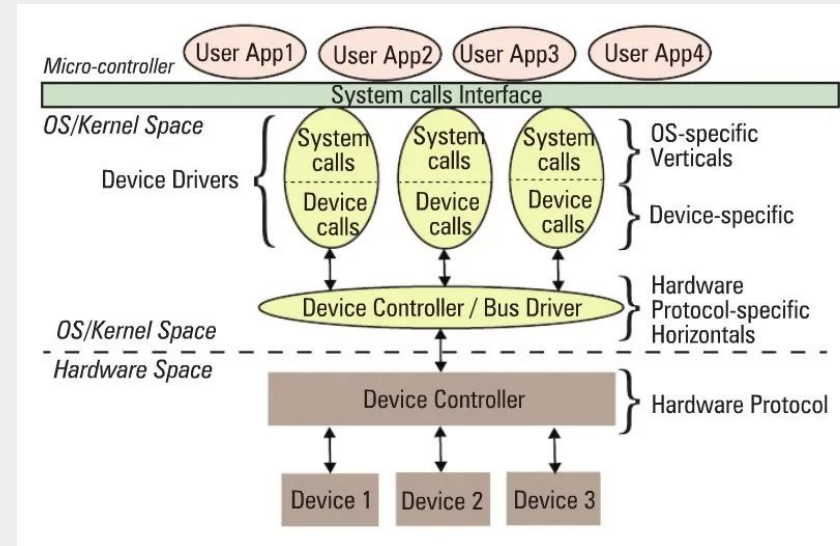
Modo de acceso a registros de E/S

- mapeado en memoria
- aislado (por instrucciones específicas)

Sistemas Operativos I - Entrada / Salida

Controladores de Dispositivos de E/S (DRIVERS):

- Conjunto de funciones que realizan Entrada o Salida (I/O) en un dispositivo.
- Suele contener código específico para manejar el dispositivo.
- Incluye funciones para leer o recibir datos, controlar el dispositivo, y manejar las interrupciones.
- El código se divide conceptualmente en dos partes:



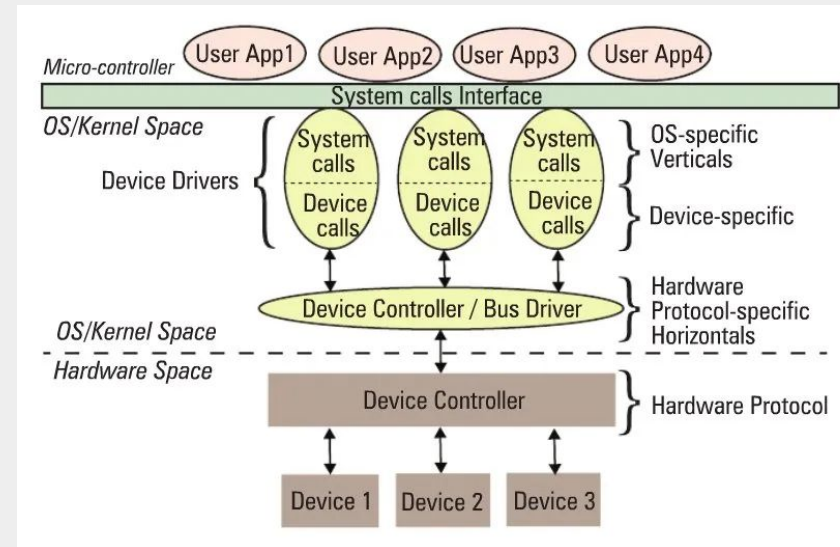
Upper half: limitada interacción con el hw (buffer)

Lower half: limitada interacción con la aplicación (buffer)

Sistemas Operativos I - Entrada / Salida

Controladores de Dispositivos de E/S (DRIVERS):

- Los procesos podrían llegar a ser bloqueados (depende de la implementación)
 - para enviar (esperar a dispositivo LISTO)
 - para recibir
- ¿Cómo coordinar procesos y drivers?



Upper half: limitada interacción con el hw (buffer)

Lower half: limitada interacción con la aplicación (buffer)

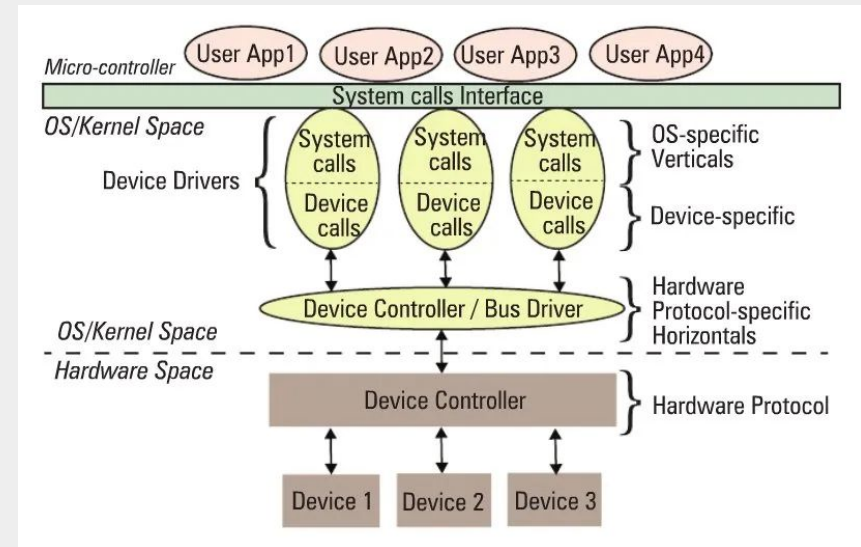
Sistemas Operativos I - Entrada / Salida

Controladores de Dispositivos de E/S (DRIVERS):

- Los procesos podrían llegar a ser bloqueados (depende de la implementación)
 - para enviar (esperar a dispositivo LISTO)
 - para recibir
- ¿Cómo coordinar procesos y drivers?

Utilizando mecanismos estándar del OS

- semáforos
- pasaje de mensajes
- resume/suspend



Upper half : limitada interacción con el hw (buffer)

Lower half: limitada interacción con la aplicación (buffer) **Cuidado: esta sección no puede bloquearse**

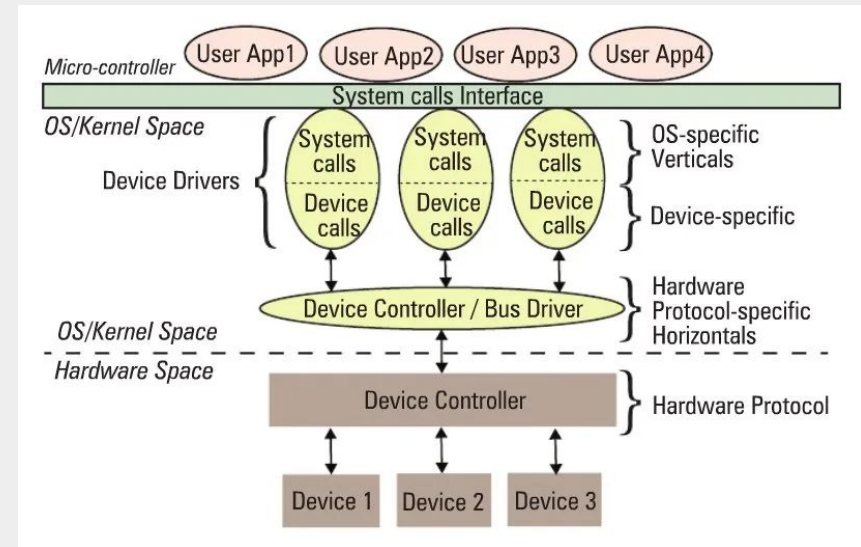
Sistemas Operativos I - Entrada / Salida

Controladores de Dispositivos de E/S (DRIVERS):

- Los procesos podrían llegar a ser bloqueados (depende de la implementación)
 - para enviar (esperar a dispositivo LISTO)
 - para recibir
- ¿Cómo coordinar procesos y drivers?

Utilizando mecanismos estándar del OS

- semáforos (Xinu)
- pasaje de mensajes
- resume/suspend



Upper half : limitada interacción con el hw (buffer)

Lower half: limitada interacción con la aplicación (buffer) **Cuidado: esta sección no puede bloquearse**

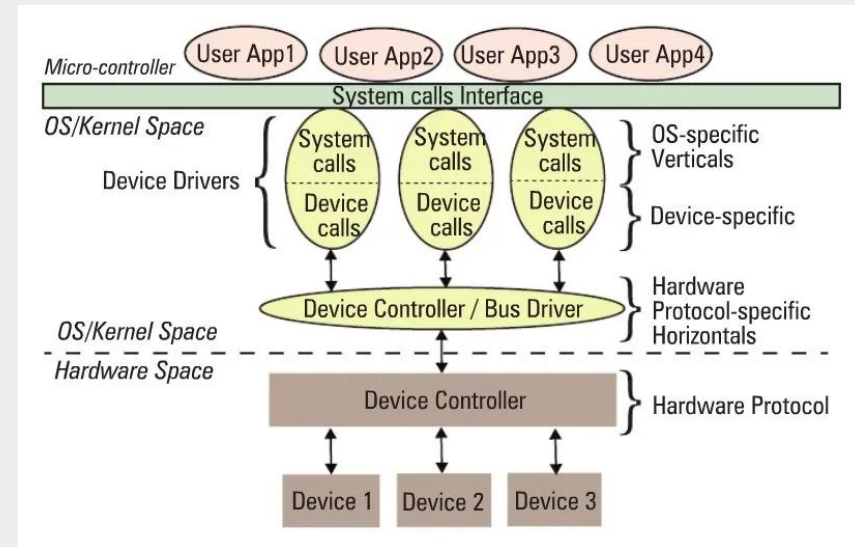
Sistemas Operativos I - Entrada / Salida

Controladores de Dispositivos de E/S (DRIVERS):

- Los procesos podrían llegar a ser bloqueados (depende de la implementación)
 - para enviar (esperar a dispositivo LISTO)
 - para recibir

Utilizando mecanismos estándar del OS

- semáforos (Xinu)
- pasaje de mensajes
- resume/suspend



Read : aplicación realiza un wait()

Write: aplicación realiza un wait() (Trick: el semáforo se interpreta como “espacio disponible”)

Lower half: limitada interacción con la aplicación (buffer) **Cuidado: esta sección no puede bloquearse**

Diseño e implementación de un controlador de hardware (driver) para un dispositivo de E/S

- Conocer las direcciones de los registros de hardware
- Programar E/S básica para conocer su funcionamiento
- Desarrollar una rutina de atención de interrupciones
- Desarrollar conjunto de rutinas útiles
 - cada driver con funciones específicas (ej. OS embebidos)
 - utilizar un paradigma general

Sistemas Operativos I - Entrada / Salida

Diseño e implementación de un controlador de hardware (driver) para un dispositivo de E/S

- Desarrollar conjunto de rutinas (interface / system calls)
 - cada driver con funciones específicas (ej. OS embebidos)
 - utilizar un paradigma general

Device	I/O paradigm
hard drive	move to a position and transfer a block of data
keyboard	accept individual characters as entered
printer	transfer an entire document to be printed
audio output	transfer a continuous stream of encoded audio
wireless network	send or receive a single network packet

Sistemas Operativos I - Entrada / Salida

Diseño e implementación de un controlador de hardware (driver) para un dispositivo de E/S

Desarrollar conjunto de rutinas (syscalls). Ejemplo UNIX (Linux, Mac OS, BSD)

- Los dispositivos se acceden mediante el sistema de archivos (ej. /dev/video0)
- Un conjunto general de funciones (**paradigma open, read, write, close**)
- **Ocultar los detalles del hardware y presenta una interfaz de alto nivel**

open()

read()

write()

close()

seek()

ioctl() (en Xinu control)

Inicialización : init(), probe() (autoconfiguración)

Uniform interfacing for device drivers
Buffering
Error reporting
Allocating and releasing dedicated devices
Providing a device-independent block size

Figure 5-13. Functions of the device-independent I/O software.

Sistemas Operativos I - Entrada / Salida

Diseño e implementación de un controlador de hardware (driver) para un dispositivo de E/S

Desarrollar conjunto de rutinas únicas (syscalls) para todos los dispositivos de E/S.

Abstracción de alto nivel.

Paradigma uniforme Ejemplo UNIX (Linux, Mac OS, BSD)

`open()`

`read()`

`write()`

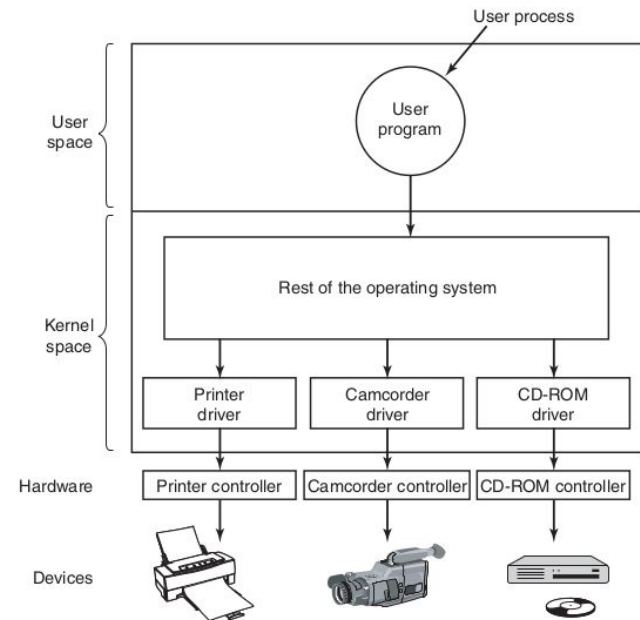
`close()`

`seek()`

`ioctl()`

(en Xinu control)

Inicialización : `init()`, `probe()` (autoconfiguración)



Sistemas Operativos I - Entrada / Salida

Diseño e implementación de un controlador de hardware
(driver) para un dispositivo de E/S

Driver con Dos secciones : upper-half y lower-half

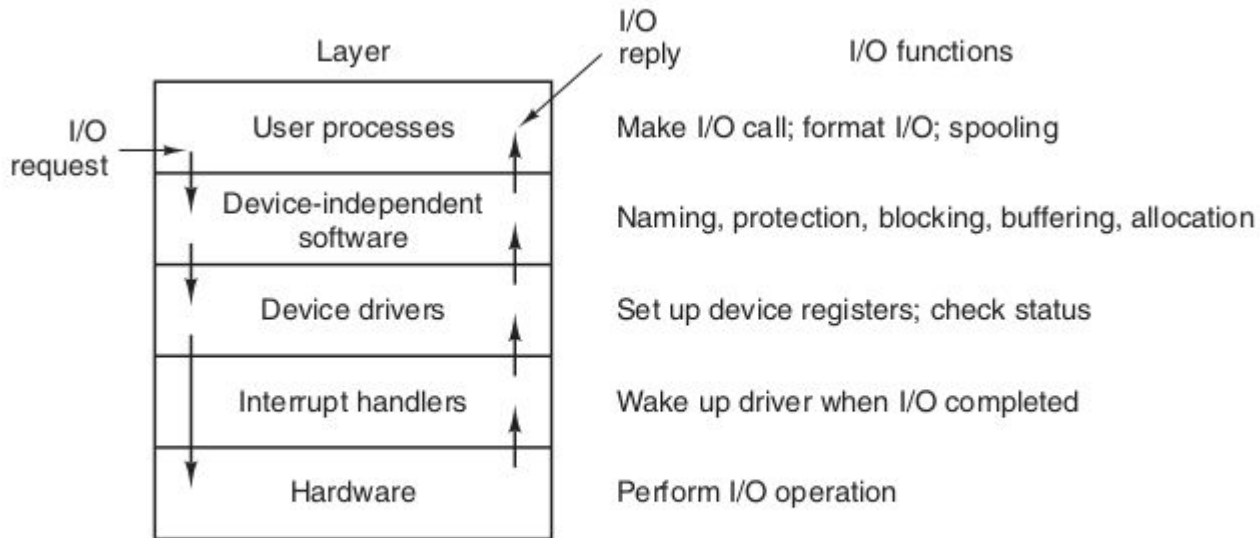


Figure 5-17. Layers of the I/O system and the main functions of each layer.

Sistemas Operativos I - Entrada / Salida

Diseño e implementación de un controlador de hardware (driver) para un dispositivo de E/S

- Se requiere una Estructura de Datos de Buffering circular (para comunicación entre el upper-half y el lower-half)
- Ejemplo de tty en Xinu
 - read(), getc(), ISR de entrada
 - write(), putc(), ISR de salida

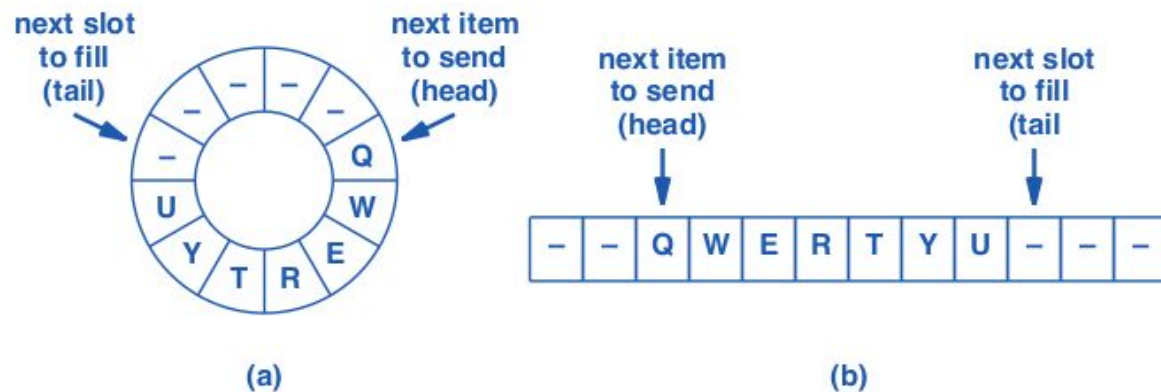


Figure 15.2 (a) A circular output buffer acting as a queue, and (b) the implementation with an array of bytes.

Sistemas Operativos I - Entrada / Salida

Ejemplo : driver tty de Xinu (/device/tty/). CONSOLE

Una aplicación que lo utiliza es el shell, para acceder a los datos de Entrada (teclado de la CONSOLA) y mostrar la Salida (pantalla de la CONSOLA).

ttygetc() - syscall del driver tty (upper-half) :

Devuelve un byte desde el buffer de entrada del driver. Bloquea si el buffer del driver está vacío.

ttyhandle_in() - ISR del driver tty (lower-half) :

Cuando arriba un byte (ej. se presiona una tecla) se ejecuta esta rutina de atención de interrupciones.

Obtiene del hw el byte y lo agrega al buffer de entrada del driver. Si el buffer está lleno lo descarta.

También señala el semáforo de sincronización entre el upper-half y el lower-half.

ttyread() - syscall del driver tty (upper-half) :

Devuelve N cantidad de bytes desde el buffer de entrada del driver.

Llama a ttygetc() N veces para obtener los N bytes.

```
/*-----  
 * ttygetc - Read one character from a tty device (interrupts disabled)  
 *-----  
 */  
devcall ttygetc(  
    struct dentry *devptr          /* Entry in device switch table */  
)  
{  
    char    ch;                    /* Character to return          */  
    struct  ttycbk *typtr;         /* Pointer to ttytab entry    */  
  
    typtr = &ttytab[devptr->dvminor];  
  
    /* Wait for a character in the buffer and extract one character */  
  
    wait(typtr->tyisem);  
    ch = *typtr->tyihead++;  
  
    /* Wrap around to beginning of buffer, if needed */  
  
    if (typtr->tyihead >= &typtr->tyibuff[TY_IBUFLEN]) {  
        typtr->tyihead = typtr->tyibuff;  
    }  
  
    /* In cooked mode, check for the EOF character */  
  
    if ( (typtr->tyimode == TY_IMCOOKED) && (typtr->tyeof) &&  
        (ch == typtr->tyeofch) ) {  
        return (devcall)EOF;  
    }  
  
    return (devcall)ch;  
}
```