TERMINAL I/O

PROJECT PRESENTATION

ITCS343
Principles of
Operating
Systems

MEMBERS

NAPHAT KHAJOHN-UDOMRITH

Introduction and Code

MANGKHALES NGAMJARUSKOTCHAKORN

Terminal I/O mode and Process

TEEKAWIN KIRDSEANG

Stty command and Summary

"THE UNIX SYSTEM IS NO EXCEPTION."

Terminal I/O Mode

CANONICAL MODE

Terminal input is processed line by line, and line editing is enabled

Lines are terminated by a newline character, generated when the user presses the enter key

A read() from the terminal returns only when a complete line of input is available, and returns at most one line

NEXT

Terminal I/O Mode

NON CANONICAL MODE

Terminal input is not gathered into lines

They can read single characters without the user needing to press the enter key

In non canonical mode no special input processing is performed

MIN == 0, TIME == 0

Polling read

POLLING READ

Returns immediately with the lesser of the number of bytes available

MIN > 0, TIME == 0

Blocking read

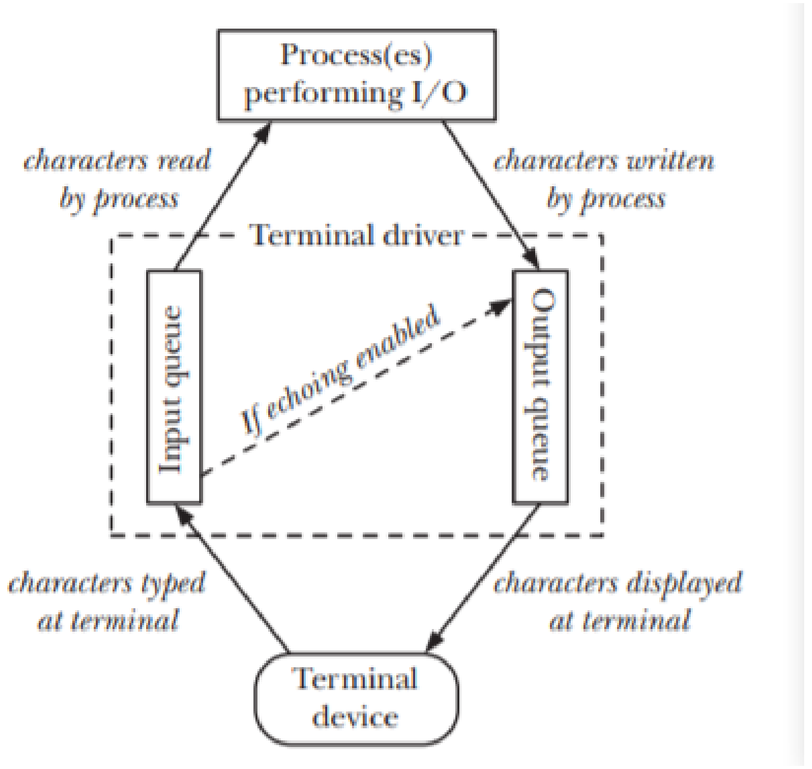
BLOCKING READ

Until the lesser of the number of bytes requested

For Example

....

Terminal I/O



Input and output queues for a terminal device

Stty Command -

BASIC CAMMAND

what is stty command

Syntax
-

stty [-F DEVICE | --file=DEVICE]
[SETTING]...
stty [-F DEVICE | --file=DEVICE] [-a|--all]

$ stty -a

speed 38400 baud; rows 25; columns 80; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = ;
eol2 = ; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread -clocal -crtscts
-ignbrk brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl ixon -ixoff
-iuclc -ixany imaxbel -iutf8
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop -
echoprt
echoctl echoke

Stty Command

-

BASIC CAMMAND

09

For Example -

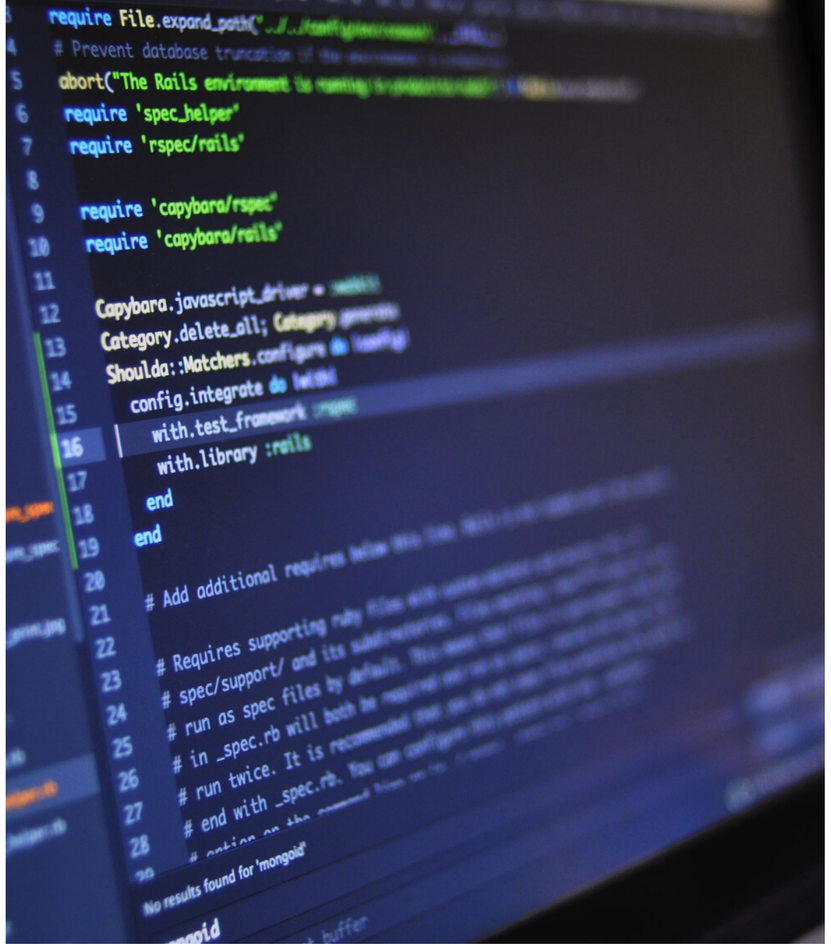
$ STTY -ECHO

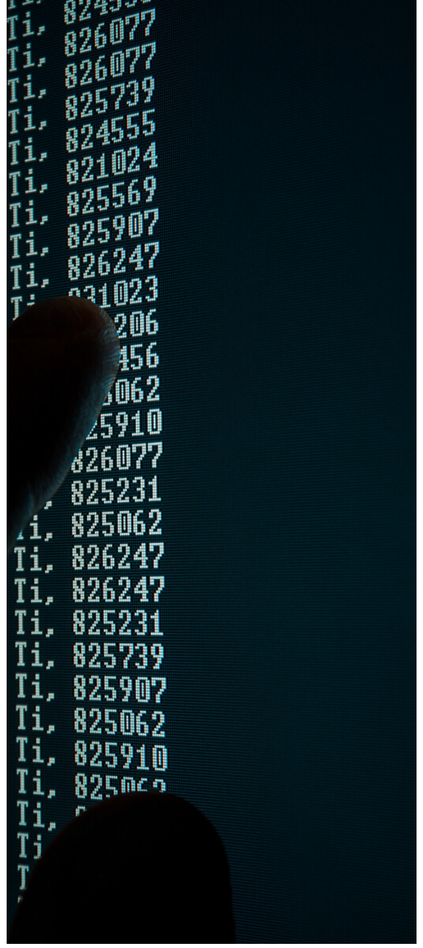
$ STTY ECHO

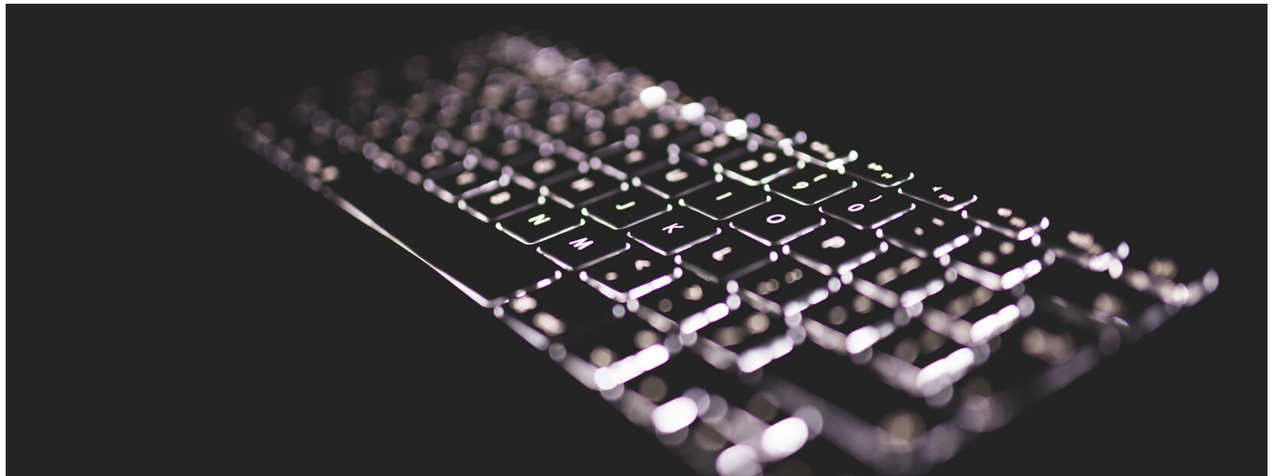
$ STTY -A

Code

Example of code







CODE EXAMPLE

```
#include "apue.h"
#include <termios.h>

int
main(void)
{
    struct termios  term;
    long            vdisable;

    if (isatty(STDIN_FILENO) == 0)
        err_quit("standard input is not a terminal device");

    if ((vdisable = fpathconf(STDIN_FILENO, _PC_VDISABLE)) < 0)
        err_quit("fpathconf error or _POSIX_VDISABLE not in effect");

    if (tcgetattr(STDIN_FILENO, &term) < 0) /* fetch tty state */
        err_sys("tcgetattr error");

    term.c_cc[VINTR] = vdisable;    /* disable INTR character */
    term.c_cc[VEOF]  = 2;           /* EOF is Control-B */

    if (tcsetattr(STDIN_FILENO, TCSAFLUSH, &term) < 0)
        err_sys("tcsetattr error");

    exit(0);
}
```

Figure 18.10 Disable interrupt character and change end-of-file character

Getting and Setting Terminal Attributes

CODE EXAMPLE

```
#include <termios.h>
int tcgetattr(int fd, struct termios *termptr);
int tcsetattr(int fd, int opt, const struct termios *termptr);
```

Both return: 0 if OK, –1 on error

Basic code

CODE EXAMPLE

```
#include <stdio.h>
int main(int argc  char* argv[] )
{
    int   ch;

    printf("Type any characters followed by the 'Enter' key.");
    printf(" Use Ctrl-D to exit.\n");

    while( ( ch = fgetc(stdin)) != EOF  )
        printf("char = '%c' code = %03d\n", ch, ch );
    return 0;
}
```

Summary

"TERMINALS HAVE MANY FEATURES AND OPTIONS, MOST OF WHICH WE'RE ABLE TO CHANGE TO SUIT YOUR NEEDS."

THANK FOR YOUR ATTENTION

THE END