

TERMINAL I/O

ITCS343
Principles of
Operating
Systems

MEMBERS

**NAPHAT KHAJOHN-
UDOMRITH**

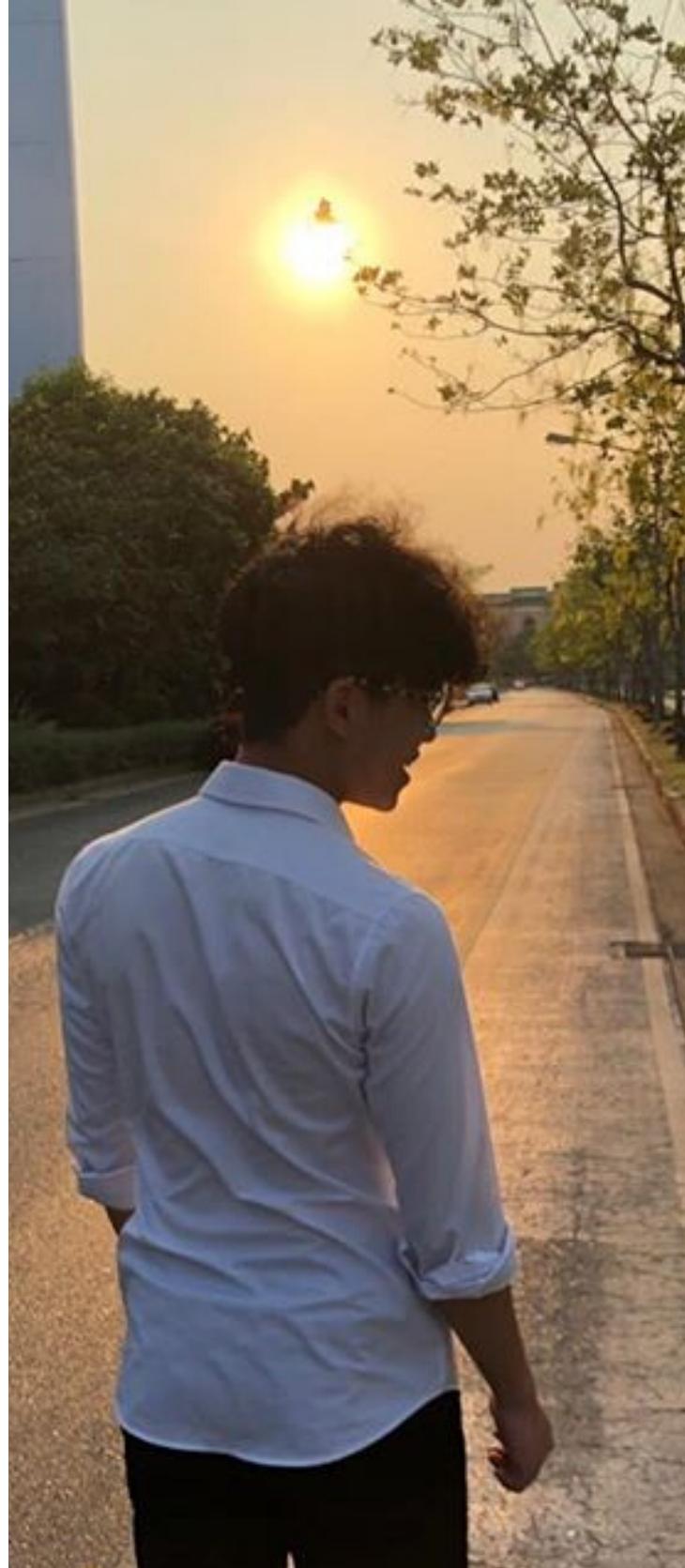
Introduction and Code

**MANGKHALES
NGAMJARUSKOTCHAKORN**

Terminal I/O mode and Process

TEEKAWIN KIRDSEANG

Stty command and Summary



Introduction

SYSTEM V, AND
THE VERSION 7

"THE UNIX SYSTEM IS
NO EXCEPTION."

1970S
03

•••

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

Terminal I/O Mode

NEXT

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

Terminal I/O Mode

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

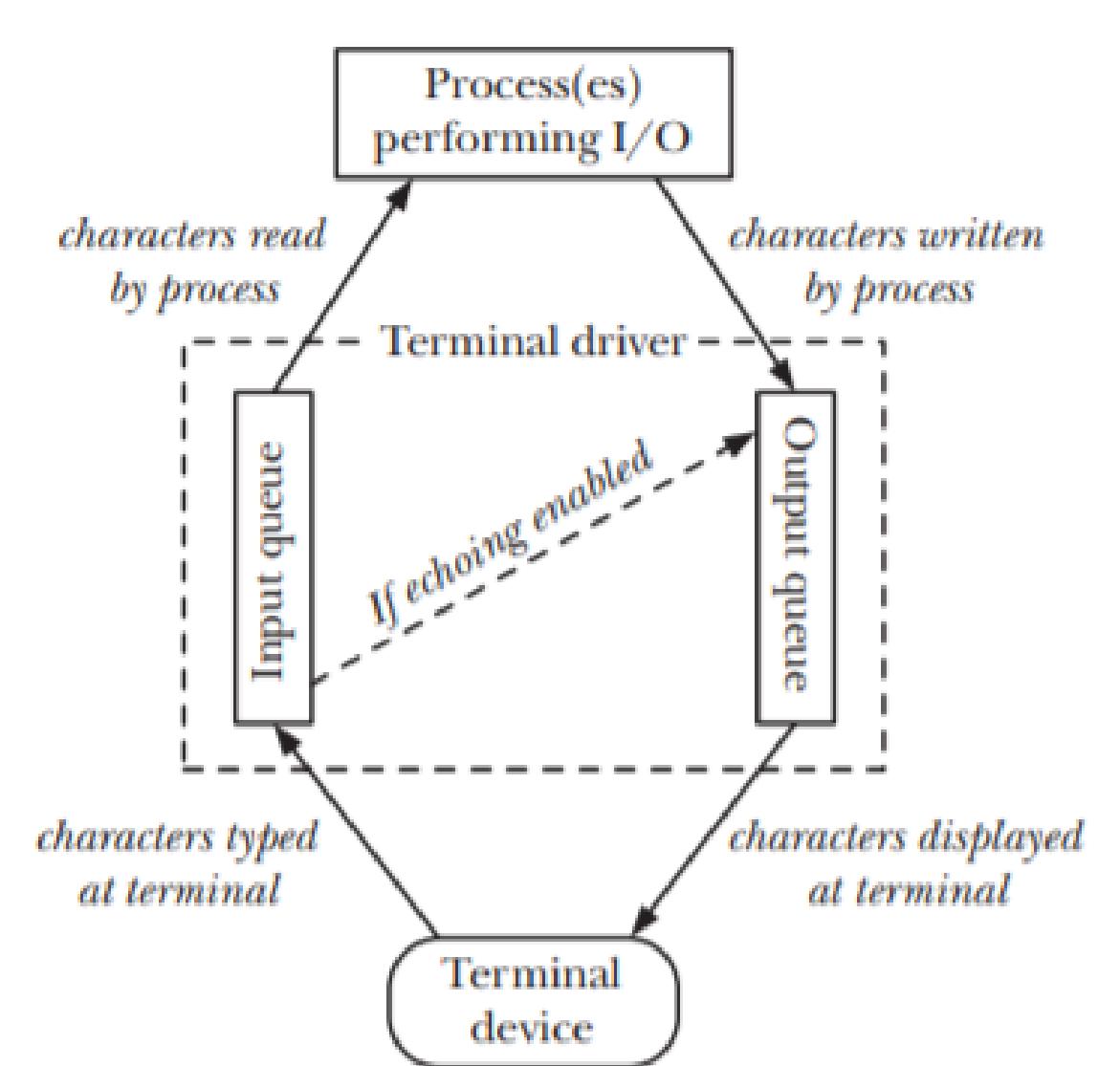
Until the lesser of the number of bytes
requested

For
Example

•••

Terminal I/O

07



Input and output queues
for a terminal device



Stty Command

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 -crtscs  
-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

\$ STTY -ECHO

\$ STTY ECHO

\$ STTY -A

For
Example

-

Code

Example of code

```
require File.expand_path('../config/environment', __FILE__)
# Prevent database truncation if the environment is test or test杠
abort("The Rails environment is running in production mode! Please run rails
      server in development mode to start a new session")
require 'spec_helper'
require 'rspec/rails'

require 'capybara/rspec'
require 'capybara/rails'

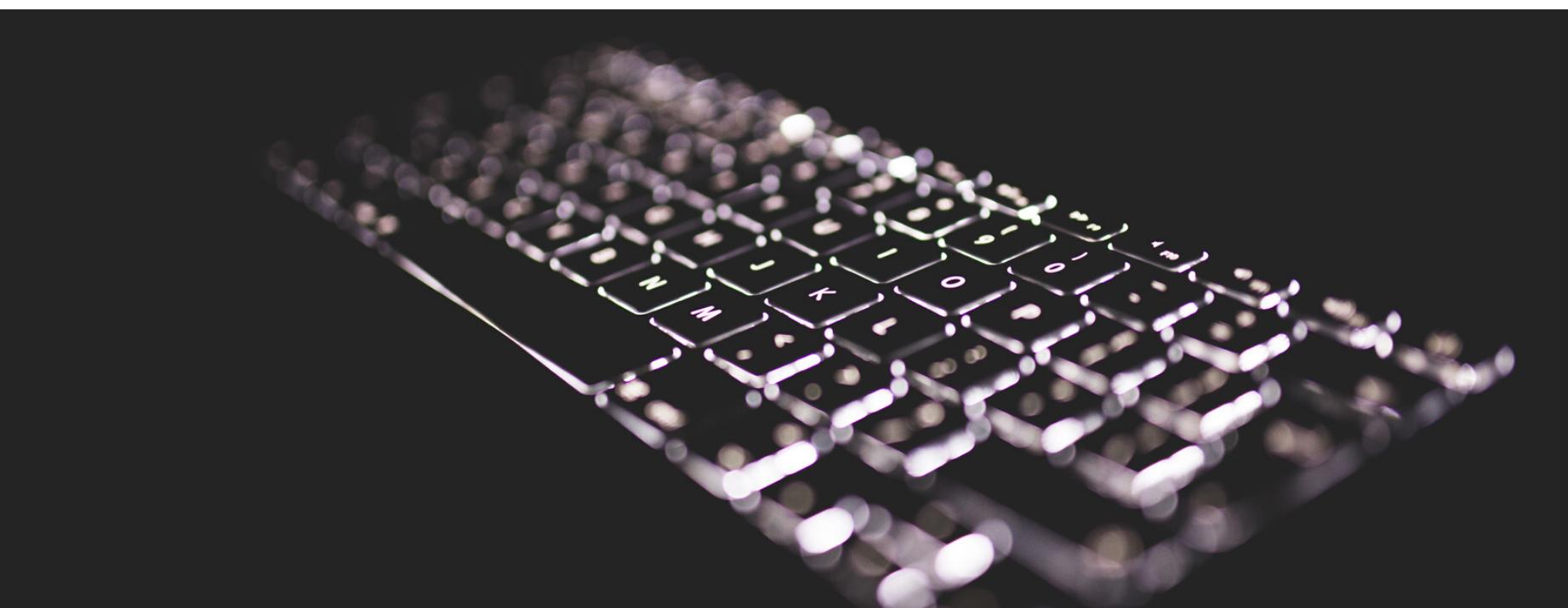
Capybara.javascript_driver = :webkit
Category.delete_all; Category.create!(name: "Ruby on Rails")
Shoulda::Matchers.configure do |config|
  config.integrate do |with|
    with.test_framework :rspec
    with.library :rails
  end
end

# Add additional requires below this line if you require them.

# Requires supporting ruby files with custom matchers and helpers
# run as spec files and its subdirectories. This means that
# in _spec.rb will both be measured and run as part of the
# tests. It is recommended that you do not name files
# end with _spec.rb. You can configure this pattern with
# --pattern in the command line or in RSpec.configure block
# in spec/spec_helper.rb

# mongoid
# mongo_mapper
# buffer
```

i, 824555
i, 826077
i, 826077
i, 825739
i, 824555
i, 821024
i, 825569
i, 825907
i, 826247
i, 821023
, 206
, 456
, 6062
, 25910
, 826077
, 825231
i, 825062
Ti, 826247
Ti, 826247
Ti, 825231
Ti, 825739
Ti, 825907
Ti, 825062
Ti, 825910
Ti, 825062
Ti, 825062
Ti, 825062



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

```
#include <termios.h>

int tcgetattr(int fd, struct termios *termptp);
int tcsetattr(int fd, int opt, const struct termios *termptp);
```

Both return: 0 if OK, -1 on error

Basic code

```
#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