1. Logger的创建

log包定义了Logger类型，该类型提供了一些格式化输出的方法。在log包中有一个预定义的“标准”logger，可以通过调用函数Print系列(Print|Printf|Println）、Fatal系列（Fatal|Fatalf|Fatalln）、和Panic系列（Panic|Panicf|Panicln）来使用。当然我们也可以自行创建一个Logger。

log标准库中提供了一个创建新logger对象的构造函数–New，支持我们创建自己的logger示例。

|  |
| --- |
| func New(out io.Writer, prefix string, flag int) \*Logger |

参数out设置日志信息写入的目的地。参数prefix会添加到生成的每一条日志前面。参数flag定义日志的属性（时间、文件等等）。

|  |
| --- |
| 举例：  func main() {  logger := log.New(os.Stdout, "<New>", log.Lshortfile|log.Ldate|log.Ltime)  logger.Println("这是自定义的logger记录的日志。") //创建的Logger也默认可以格式化输出方法  } |

运行结果如下：(os.Stdout指定了将logger输出到屏幕上)

|  |
| --- |
| <New>2022/02/14 21:34:36 main.go:11: 这是自定义的logger记录的日志。 |

1. Flag—日志属性

log标准库提供了如下的flag选项，它们是一系列定义好的常量。

|  |
| --- |
| const (  // 控制输出日志信息的细节，不能控制输出的顺序和格式。  // 输出的日志在每一项后会有一个冒号分隔：例如2009/01/23 01:23:23.123123 /a/b/c/d.go:23: message  Ldate = 1 << iota // 输出日期，如2009/01/23  Ltime // 输出时间，如01:23:23  Lmicroseconds // 输出微秒级别的时间，如01:23:23.123123（用于增强Ltime位）  Llongfile // 输出文件全路径名+行号： /a/b/c/d.go:23  Lshortfile // 输出文件名+行号：d.go:23（会覆盖掉Llongfile）  LUTC // 输出使用UTC时间  LstdFlags = Ldate | Ltime // 标准logger的初始值  )  上述常量的值分别为1,2,4,8,16,32,3 |

1. 修改创建Logeer时的三项形参信息
2. 修改参数3，logeer的flag

log标准库中的Flags函数会返回标准logger的输出配置，而SetFlags函数用来设置标准logger的输出配置。

|  |
| --- |
| func Flags() int  func SetFlags(flag int) |

使用事例：

|  |
| --- |
| func main() {  log.SetFlags(log.Llongfile | log.Lmicroseconds | log.Ldate)  log.Println("这是一条很普通的日志。")  } |

结果显示：

|  |
| --- |
| 2022/02/14 21:26:42.546952 D:/Golang/src/go\_code/project02/log/test/main.go:9: 这是一条很普通的日志。 |

1. 修改参数2，logger前缀

log标准库中还提供了关于日志信息前缀的两个方法：

|  |
| --- |
| func Prefix() string  func SetPrefix(prefix string) |

其中Prefix函数用来查看标准logger的输出前缀，SetPrefix函数用来设置输出前缀。

使用事例：

|  |
| --- |
| func main() {  log.SetFlags(log.Llongfile | log.Lmicroseconds | log.Ldate)  log.Println("这是一条很普通的日志。")  log.SetPrefix("[小王子]")  log.Println("这是一条很普通的日志。")  } |

结果显示：

|  |
| --- |
| [小王子]2022/02/14 21:29:01.392368 D:/Golang/src/go\_code/project02/log/test/main.go:11: 这是一条很普通的日志。 |

为我们的日志信息添加指定的前缀，方便之后对日志信息进行检索和处理。

1. 参数1，logger的输出目标

SetOutput函数用来设置标准logger的输出目的地，默认是标准错误输出。

|  |
| --- |
| func SetOutput(w io.Writer) |

例如，下面的代码会把日志输出到同目录下的xx.log文件中。

|  |
| --- |
| func main() {  logFile, err := os.OpenFile("./xx.log", os.O\_CREATE|os.O\_WRONLY|os.O\_APPEND, 0644)  if err != nil {  fmt.Println("open log file failed, err:", err)  return  }  log.SetOutput(logFile)  log.SetFlags(log.Llongfile | log.Lmicroseconds | log.Ldate)  log.Println("这是一条很普通的日志。")  log.SetPrefix("[小王子]")  log.Println("这是一条很普通的日志。")  } |

如果要使用标准的logger，我们通常会把上面的配置操作写到init函数中。

|  |
| --- |
| func init() {  logFile, err := os.OpenFile("./xx.log", os.O\_CREATE|os.O\_WRONLY|os.O\_APPEND, 0644)  if err != nil {  fmt.Println("open log file failed, err:", err)  return  }  log.SetOutput(logFile)  log.SetFlags(log.Llongfile | log.Lmicroseconds | log.Ldate)  } |