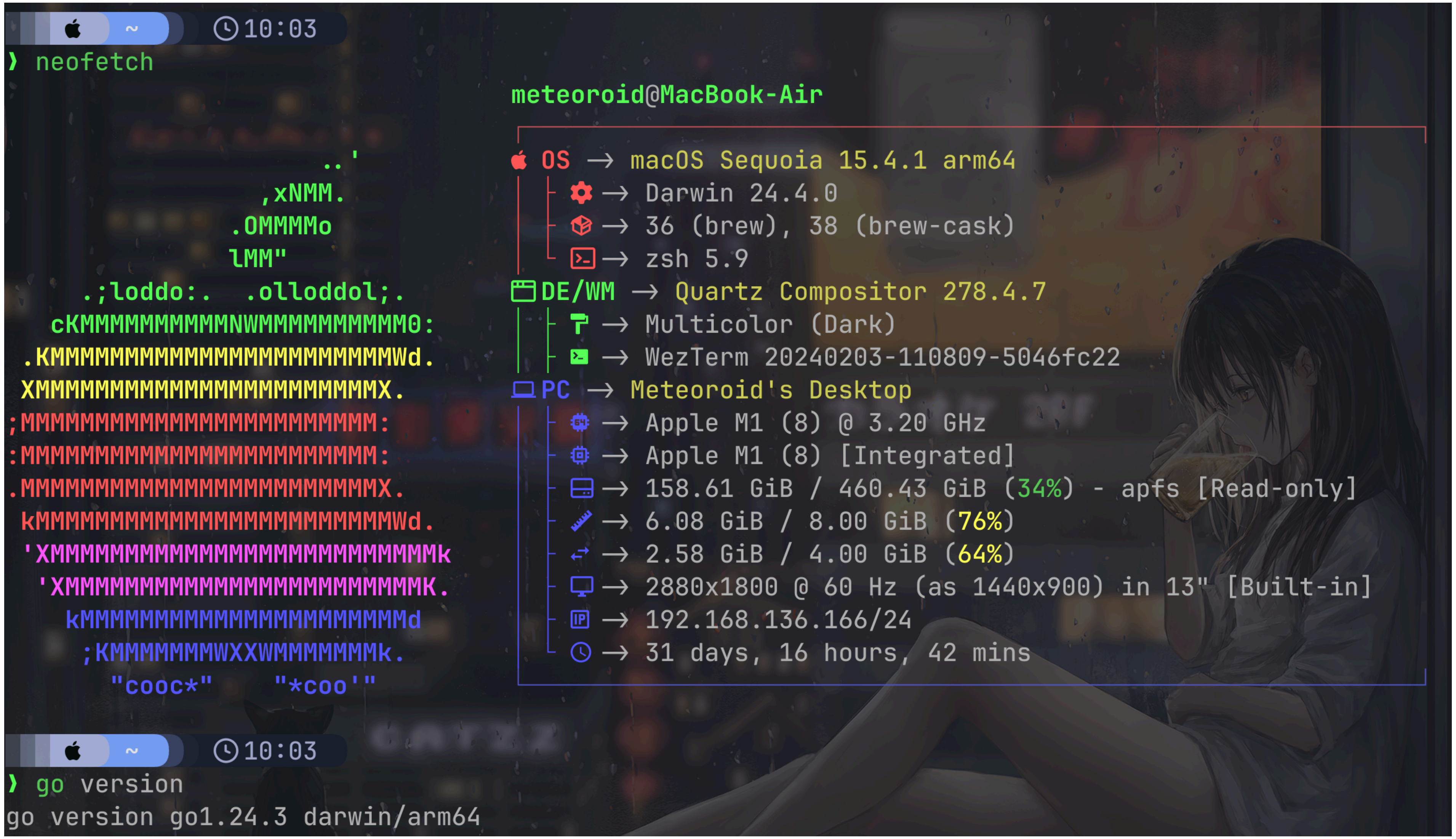
多媒體安全 - 期末報告

電通三乙

C111110233 余明耀

系統環境介紹



```
 ~  10:03
❯ neofetch
                                  meteoroid@MacBook-Air

                     ..'          OS → macOS Sequoia 15.4.1 arm64
                 ,xNMM.           │ ├ → Darwin 24.4.0
               .OMMMMo            │ ├ → 36 (brew), 38 (brew-cask)
               lMM"               │ └ → zsh 5.9
     .;loddo:.  .olloddol;.       DE/WM → Quartz Compositor 278.4.7
   cKMMMMMMMMMMNWMMMMMMMMMM0:     │ ├ → Multicolor (Dark)
 .KMMMMMMMMMMMMMMMMMMMMMMMWd.     │ ├ → WezTerm 20240203-110809-5046fc22
 XMMMMMMMMMMMMMMMMMMMMMMMX.       PC → Meteoroid's Desktop
;MMMMMMMMMMMMMMMMMMMMMMMM:        │ ├ → Apple M1 (8) @ 3.20 GHz
:MMMMMMMMMMMMMMMMMMMMMMMM:        │ ├ → Apple M1 (8) [Integrated]
.MMMMMMMMMMMMMMMMMMMMMMMMX.       │ ├ → 158.61 GiB / 460.43 GiB (34%) - apfs [Read-only]
 kMMMMMMMMMMMMMMMMMMMMMMMMWd.     │ ├ → 6.08 GiB / 8.00 GiB (76%)
 'XMMMMMMMMMMMMMMMMMMMMMMMMMMk    │ ├ → 2.58 GiB / 4.00 GiB (64%)
  'XMMMMMMMMMMMMMMMMMMMMMMMMK.    │ ├ → 2880x1800 @ 60 Hz (as 1440x900) in 13" [Built-in]
    kMMMMMMMMMMMMMMMMMMMMMMd      │ ├ → 192.168.136.166/24
     ;KMMMMMMMWXXWMMMMMMMk.       │ └ → 31 days, 16 hours, 42 mins
       "cooc*"    "*coo'"

 ~  10:03
❯ go version
go version go1.24.3 darwin/arm64
```

專案結構

```
secretmerger
├── cmd
│   ├── main
│   │   └── main.go
│   └── test
│       └── main.go
├── go.mod
├── internal
│   ├── imageio
│   │   ├── bmp.go
│   │   └── utils.go
│   └── process
│       ├── combine.go
│       ├── psnr.go
│       └── reverse.go
├── output
│   ├── outputXReversed.bmp
│   ├── Z_combined.bmp
│   ├── Z_double_reversed.bmp
│   └── Z_reversed.bmp
└── testdata
    ├── baboon_gray.bmp
    ├── F16_gray.bmp
    ├── Lena_gray.bmp
    └── peppers_gray.bmp
```

位元反轉 `internal/process/reverse.go`

```go
// ReverseByte 反轉一個 byte 的 bit 順序
func ReverseByte(b byte) byte {
	var r byte = 0
	for i := 0; i < 8; i++ {
		r <<= 1 // r 左移
		r |= b & 1 // 取 b 最右邊的 bit (b & 1) 放到 r 最右邊的 bit (r |= ...)
		b >>= 1 // b 右移
	}
	return r
}
// ReverseBits 反轉一個 byte 的 bit 順序，並保存成 B4B5B6B7 0000
func ReverseBits(b byte) byte {
	var r byte = 0
	for i:= 0; i < 8; i++ {
		r <<= 1
		r |= b & 1
		b >>= 1
	}
	r <<= 4
	return r
}
```

更改整張圖片 `internal/process/reverse.go`

```go
// ReverseImageBits 對每個像素的 bit 順序做反轉（僅支援灰階圖）
func ReverseImageBits(img *image.Gray) *image.Gray {
	bounds := img.Bounds()
	newImg := image.NewGray(bounds)

	for y := bounds.Min.Y; y < bounds.Max.Y; y++ {
		for x := bounds.Min.X; x < bounds.Max.X; x++ {
			orig := img.GrayAt(x, y).Y
			newVal := ReverseByte(orig)
			newImg.SetGray(x, y, color.Gray{Y: newVal})
		}
	}
	return newImg
}
```

組合圖片 `internal/process/combine.go`

```go
// Combine 將兩張灰階影像 X 和 Y 合併為 Z
func Combine(x, y *image.Gray) *image.Gray {
    if x.Bounds() != y.Bounds() {
        panic("images must have the same dimensions")
    }

    bounds := x.Bounds()
    z := image.NewGray(bounds)

    for yCoord := bounds.Min.Y; yCoord < bounds.Max.Y; yCoord++ {
        for xCoord := bounds.Min.X; xCoord < bounds.Max.X; xCoord++ {
            xPixel := x.GrayAt(xCoord, yCoord).Y
            yPixel := y.GrayAt(xCoord, yCoord).Y
            yPixel = ReverseBits(yPixel)

            // 合併
            zPixel := (xPixel & 0xF0) | (yPixel >> 4)
            z.SetGray(xCoord, yCoord, color.Gray{Y: zPixel})
        }
    }

    return z
}
```

計算 PSNR `internal/process/psnr.go`

讀取圖片 `internal/imageio/bmp.go`

```go
const bmpHeaderSize = 54 // BMP header size

func ReadGrayBMP(path string) ([]byte, []byte, error) {
	file, err := os.ReadFile(path)
	if err != nil {
		return nil, nil, err
	}

	if len(file) < bmpHeaderSize {
		return nil, nil, fmt.Errorf("not a valid BMP file")
	}

	header := file[:bmpHeaderSize]
	pixels := file[bmpHeaderSize:]
	return header, pixels, nil
}
```

主程式邏輯 `cmd/main/main.go`

```go
func main() {
	inputY := "testdata/peppers_gray.bmp"
	inputX := "testdata/baboon_gray.bmp"
	outputZ := "output/Z_combined.bmp"
	outputReversed := "output/Z_reversed.bmp"
	outputDoubleReversed := "output/Z_double_reversed.bmp"
	// 讀取 X 和 Y 影像
	headerX, pixelsX, err := imageio.ReadGrayBMP(inputX)
	if err != nil {
		panic(err)
	}
	headerY, pixelsY, err := imageio.ReadGrayBMP(inputY)
	if err != nil {
		panic(err)
	}

	// 確保兩張影像大小相同
	widthX := int(headerX[18]) | int(headerX[19])<<8 | int(headerX[20])<<16 | int(headerX[21])<<24
	heightX := int(headerX[22]) | int(headerX[23])<<8 | int(headerX[24])<<16 | int(headerX[25])<<24
	widthY := int(headerY[18]) | int(headerY[19])<<8 | int(headerY[20])<<16 | int(headerY[21])<<24
	heightY := int(headerY[22]) | int(headerY[23])<<8 | int(headerY[24])<<16 | int(headerY[25])<<24

	if widthX != widthY || heightX != heightY {
		panic("Images X and Y must have the same dimensions")
	}

	// 將像素轉換為 *image.Gray
	imgX := &image.Gray{
		Pix:    pixelsX,
		Stride: widthX,
		Rect:   image.Rect(0, 0, widthX, heightX),
	}
	imgY := &image.Gray{
		Pix:    pixelsY,
		Stride: widthY,
		Rect:   image.Rect(0, 0, widthY, heightY),
	}
}
```

主程式邏輯(續) `cmd/main/main.go`

```go
func main() {
	// 合併 X 和 Y 成 Z
	imgZ := process.Combine(imgX, imgY)

	// 對 Z 進行一次反轉
	reversedZ := process.ReverseImageBits(imgZ)

	// 再次反轉 Z
	doubleReversedZ := process.ReverseImageBits(reversedZ)

	// 保存 Z 、反轉後的 Z 和兩次反轉的 Z
	if err := imageio.WriteGrayBMP(outputZ, headerX, imgZ.Pix); err != nil {
		panic(err)
	}
	if err := imageio.WriteGrayBMP(outputReversed, headerX, reversedZ.Pix); err != nil {
		panic(err)
	}
	if err := imageio.WriteGrayBMP(outputDoubleReversed, headerX, doubleReversedZ.Pix); err != nil {
		panic(err)
	}

	fmt.Println("Saved combined image to", outputZ)
	fmt.Println("Saved reversed image to", outputReversed)
	fmt.Println("Saved double-reversed image to", outputDoubleReversed)
	fmt.Println("All images saved successfully.")
}
```