|  |
| --- |
| **약[514126] 컴퓨터네트워크** |
| **실습 #05 문제 및 보고서** |

|  |  |
| --- | --- |
| **이름** | 이인호 |
| **학번** | 20165326 |
| **소속**  **학과/대학** | 스마트 IOT |
| **분반** | 01 (담당교수: 김태운) |

## <주의사항>

* 개별 과제 입니다. (팀으로 진행하는 과제가 아니며, 모든 학생이 보고서를 제출해야 함)
* 각각의 문제 바로 아래에 답을 작성 후 제출해 주세요.
  + 소스코드/스크립트 등을 작성한 경우, 해당 파일의 이름도 적어주세요.
* SmartLEAD 제출 데드라인: 실습 요일별로 데드라인 다름!!
  + 화요일 분반 : 다음 실습날짜 전날까지
  + 목요일 분반 : 다음 실습날짜 전날까지
  + 데드라인을 지나서 제출하면 24시간 단위로 20%씩 감점(5일 경과 시 0점)
  + 주말/휴일/학교행사 등 모든 날짜 카운트함
  + 부정행위 적발 시, 원본(보여준 사람)과 복사본(베낀 사람) 모두 0점 처리함
  + 예외 없음
* SmartLEAD 에 아래의 파일을 제출해 주세요
  + 보고서(PDF 파일로 변환 후 제출을 권장하나, WORD로 제출해도 됨)
  + 보고서 파일명에 이름과 학번을 입력해 주세요.
  + 소스코드, 스크립트, Makefile 등을 작성해야 하는 경우, 모든 파일을 하나의 zip 파일로 압축하여 제출

## <개요>

이번 과제는 리눅스 프로그래밍입니다. 리눅스에서 프로그래밍 하기 위해 필요한 각종 Tool 사용법, 그리고 C 언어 프로그래밍 복습에 관련된 내용으로 구성되어 있습니다.

## <실습 과제>

|  |
| --- |
| **[Q 0] 요약 [배점: 10]**  이번 과제에서 배운 내용 또는 과제 완성을 위해서 무엇을 했는지 2~3문장으로 요약하세요. |

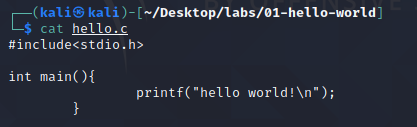
답변:

이번 과제를 하면서 텍스트 에디터를 활용해서 프로그래밍을 하는 법을 배웠고, 프로그래밍한 코드를 컴파일하는 다양한방법과 전에 배운 라우팅이라는 이론을 프로그래밍으로 배울 수 있었다.

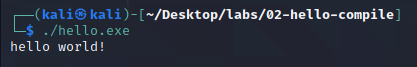
|  |
| --- |
| **[Q 1] 텍스트 에디터 [배점: 10]**  텍스트 에디터는 소스코드를 작성하기 위해 반드시 필요한 프로그램이며, 리눅스 OS에서 사용할 수 있는 다양한 텍스트 에디터가 있습니다. GUI 지원 여부에 따라서 command-line (CL) text editor 와 GUI text editor로 분류할 수 있습니다. Kali 에 기본적으로 설치된 CL 에디터로는 vi (또는 vim), nano 등이 있고, GUI 에디터로는 mousepad 등이 있습니다.  CL 에디터 중에서는 vim을 추천합니다. 익숙해지는데 시간이 걸리긴 하지만 프로그램이 가볍고 아주 강력합니다. GUI 에디터 중에서는 sublime text, visual studio code 등을 추천합니다. 본인이 선호하는 텍스트 에디터를 사용하세요. 어떤 것이든 상관 없습니다.  **문제 1)** 바탕화면에서 labs 라는 디렉토리를 만들고, 그 안에 01-hello-world 라는 디렉토리를 만드세요. 터미널 화면에서 01-hello-world 디렉토리까지 이동한 후 $pwd를 입력하고, 터미널 화면을 캡처하여 아래에 첨부하세요. (pwd는 현재 디렉토리 경로를 출력하는 명령어 입니다)  **문제 2)** hello.c 라는 텍스트 파일을 만들고, “Hello world!\n”를 출력하는 소스코드를 작성하세요. 터미널 화면에서 $cat hello.c 를 입력하고, 터미널 화면을 캡처하여 아래에 첨부하세요. |

답변 1) 

답변 2)

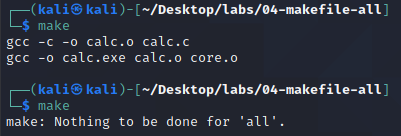


|  |
| --- |
| **[Q 2] 소스코드 컴파일 [배점: 10]**  C언어로 작성된 소스코드를 컴파일 하고, 실행 가능한 파일을 생성해 보겠습니다. 터미널 프로그램에서 01-hello-world 디렉토리까지 이동한 후, 아래와 같이 입력하세요  $cd .. // 상위 디렉토리로 이동  $cp 01-hello-world 02-hello-compile -r // 디렉토리 전체 복사  $cd 02-hello-compile // 02 디렉토리로 이동  소스코드를 컴파일 하는 방법은 다음과 같습니다.  $ls (컴파일 전에 디렉토리에 저장된 파일 목록 확인)  $gcc hello.c -o hello.exe (hello.c를 컴파일 하여 hello.exe 라는 실행파일 만들기[[1]](#footnote-1))  $ls (컴파일 후에 디렉토리에 저장된 파일 목록 확인)  ‘hello.exe’라는 실행파일이 만들어 졌습니다. 이 파일을 실행하는 방법은 $./hello.exe 라고 입력하면 됩니다.  **문제 1)** hello.exe 파일을 실행하고, 터미널 화면을 캡처하여 아래에 첨부하세요.  **문제 2)** 컴파일 시 -o hello.exe 라는 옵션 때문에 실행 파일을 hello.exe 라는 이름으로 생성합니다. $gcc hello.c 라고만 입력해도 실행파일이 생성되는데, 이 때 생성되는 실행파일의 기본 이름은 무엇인가요? |

답변 1) 

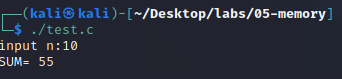
답변 2)a.out

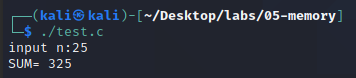
|  |
| --- |
| **[Q 3] Makefile [배점: 10]**  Makefile은 컴파일을 자동화해주는 도구입니다. 여러 개의 소스코드로 구성된 프로젝트의 경우 특히나 유용합니다. 최근에 수정된 소스 코드만 컴파일 해 주는 기능을 가지고 있어서, 대규모 프로젝트의 경우 컴파일 시간을 절약하는데 도움이 됩니다. 그리고, 컴파일 명령이 복잡한 경우, 복잡한 컴파일 명령을 매번 입력하지 않고도 컴파일 할 수 있어서 유용합니다.  ‘labs’ 디렉토리 아래에 ’03-makefile’ 디렉토리를 생성하고, 그 아래에 다음과 같이 두개의 소스 코드를 생성하세요.    두개의 소스코드를 동시에 컴파일 하는 방법은: $gcc calc.c core.c -o calc.exe 입니다. 매번 컴파일 할 때 마다 위의 명령어를 입력하는 대신, Makefile을 이용하면 간편하게 컴파일 할 수 있습니다. Makefile 이라는 파일을 만들고, 아래와 같이 입력하세요.    터미널 화면에서 $make 라고 입력하면 아래와 같이 자동을 컴파일을 수행합니다.    실행파일을 삭제하고자 하면 $make clean을 입력하세요.    Makefile을 제대로 만들어봅시다. 03-makefile 디렉토리를 복사해서 04-makefile-all 디렉토리를 만들고, Makefile을 아래와 같이 수정하세요.    **문제 1)** $make 를 입력해서 빌드하세요. 그리고, 다시 한번 make 를 입력하면 어떤 메시지가 나오나요? 왜 이런 메시지가 나오나요?  **문제 2)** core.c 파일을 열고 return a+b;를 return a+b+0;로 수정한 뒤 다시 $make를 입력하세요. 모든 소스코드 (calc.c, core.c)가 재컴파일 되었나요? 예/아니오로 답하고, 그 이유는 무엇인지 설명하세요. |

답변 1)  변경된것이 없기 때문에

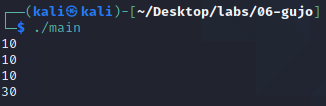
답변 2) 예 core.c 파일이 수정되었기 때문이다.

|  |
| --- |
| **[Q 4] 동적 메모리 [배점: 10]**  아래와 같이 동작하는 프로그램을 작성하세요. ‘malloc’ 또는 ‘calloc’ 을 사용하고, main에서 return 직전에 메모리 공간을 free 하세요.   1. 터미널에서 사용자로부터 정수를 입력 받음 (입력 받은 정수를 n 이라고 하겠습니다) 2. 사용자가 입력한 n을 크기로 하는 정수 배열을 만들고, 배열에 1 부터 n 까지 순차적으로 저장 3. 배열의 처음부터 끝까지 저장된 숫자의 합을 계산하는 반복문을 작성하고, 합을 sum 이라는 int 변수에 저장 4. 변수 sum에 저장된 숫자를 출력   예를 들어, 사용자가 5라는 숫자를 입력하면 크기가 5인 정수 배열을 만들고, 1,2,3,4,5를 배열에 저장하고, 반복문을 이용해서 1부터 5까지 합을 계산한 후, 그 결과인 15를 출력함.  **문제 1)** 10을 입력으로 해서 프로그램을 실행하고, 터미널 화면을 캡처해서 아래에 붙여 넣으세요.  **문제 2)** 25를 입력으로 해서 프로그램을 실행하고, 터미널 화면을 캡처해서 아래에 붙여 넣으세요. |

답변 1) 

답변 2) 

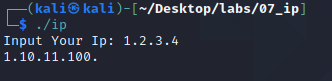
|  |
| --- |
| **[Q 5] 구조체 [배점: 10]**  struct info {int n;}; 라는 구조체를 사용하는 main.c를 작성하세요.  ‘main.c’ 코드를 아래와 같이 코딩하세요. 먼저, main.c코드에서 main 함수 밖에 아래와 같이 두 개의 함수를 정의하세요.   1. [main 함수 밖에서…] callByVal 이라는 함수를 정의하고, 함수 내에서 info 구조체의 n 값을 20으로 설정. 이 함수는 call by value를 이용해서 info 구조체를 전달받음. 2. [main 함수 밖에서…] callByRef 라는 함수를 정의하고, 함수 내에서 info 구조체의 n 값을 30으로 설정. 이 함수는 call by reference를 이용해서 info 구조체를 전달받음   ‘main’ 함수를 아래와 같이 정의하세요.   1. struct info 구조체 선언 : struct info myinfo; 2. myinfo.n=10; 으로 변수 n의 값을 설정한 후, myinfo.n 정수값을 터미널에 출력 3. callByVal에 myinfo 전달하고, callByVal에서 리턴하면 myinfo.n 정수값을 터미널에 출력 4. myinfo.n=10; 으로 변수 n의 값을 설정한 후, myinfo.n 정수값을 터미널에 출력 5. callByRef에 myinfo 전달하고, callByRef에서 리턴하면 myinfo.n 정수값을 터미널에 출력   **문제 1)** main.c를 컴파일 후 실행하고, 터미널 화면을 캡처하여 아래에 첨부하세요.  **문제 2)** 위의 4번 과정에서 출력한 myinfo.n과 5번 과정에서 출력한 myinfo.n의 값이 같습니다. 이유는 무엇인가요?  **문제 3)** 위의 6번 과정에서 출력한 myinfo.n과 7번 과정에서 출력한 myinfo.n의 값이 다릅니다. 이유는 무엇인가요? |

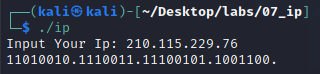
답변 1) 

답변 2)call by value는 변수의 값만 불러오는 것이라서 값이 같게 나온다.

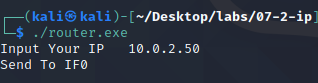
답변 3) call by reference는 주소 값을 불러와서 변수를 변경하는 것이기 때문에 값이 다르게 나온다.

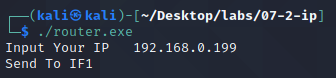
|  |
| --- |
| **[Q 6] IP 주소 변환 :10진수 => 2진수 [배점: 10]**  아래와 같이 동작하는 프로그램을 작성하세요.   1. 터미널에서 사용자로부터 IP 주소를 입력 받음. 사용자는 1.2.3.4와 같은 방식으로 IP 주소를 입력함 (즉, 10진수 숫자 4개를 3개의 점으로 구분한 형태의 문자열로 입력) 2. 입력 받은 IP 주소를 32비트 이진수로 변환해서 아래와 같은 형식으로 화면에 출력. 예를 들어, 사용자가 1.2.3.4라고 IP 를 입력한 경우, 출력은 다음과 같음 00000001.00000010.00000011.00000100   **문제1)** 사용자가 1.2.3.4를 입력한 경우, 그 결과(터미널 화면)를 캡처해서 아래에 첨부  **문제 2)** 사용자가 210.115.229.76을 입력한 경우, 그 결과(터미널 화면)를 캡처해서 아래에 첨부 |

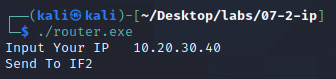
답변 1) 

답변 2) 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **[Q 7] 라우팅 & 비트마스킹 [배점: 10]**  3개의 출력 포트를 가진 라우터가 있고, 라우터는 아래와 같은 라우팅 규칙을 가지고 있다.   |  |  |  |  | | --- | --- | --- | --- | | 우선순위 | 목적지 | 서브넷 마스크 | 인터페이스(=출력 포트) | | 1 | 10.0.2.0 | 255.255.255.0 | IF0 | | 2 | 192.168.0.0 | 255.255.0.0 | IF1 | | 3 | 0.0.0.0 | 0.0.0.0 | IF2 |   위의 라우팅 테이블을 기반으로, 목적지 주소가 입력되면 어떤 인터페이스를 사용할지를 출력하는 프로그램 router.c를 작성하시오. 동작 방식은 다음과 같다.(참고: 아래에서 AND는 비트 연산자 AND를 의미함)  목적지 주소가 aaa.bbb.ccc.ddd인 IP 주소가 입력으로 주어지면 (주소는 터미널에서 사용자 입력으로 받음),  IF( aaa.bbb.ccc.ddd AND 255.255.255.0 == 10.0.2.0 ) PRINT(“Send to IF0);  ELSE-IF (aaa.bbb.ccc.ddd and 255.255.0.0 == 192.168.0.0 ) PRINT(“Send to IF1);  ELSE-IF (aaa.bbb.ccc.ddd and 0.0.0.0 == 0.0.0.0) PRINT(“Send to IF2);  ELSE PRINT("Error");  위와 같이 동작하는 router.c를 코딩하세요.  **문제 1)** 사용자 입력이 10.0.2.50 인 경우, 그 결과(터미널 화면)를 캡처하여 아래에 첨부  **문제 2)** 사용자 입력이 192.168.0.199 인 경우, 그 결과(터미널 화면)를 캡처하여 아래에 첨부  **문제 3)** 사용자 입력이 10.20.30.40 인 경우, 그 결과(터미널 화면)를 캡처하여 아래에 첨부 |

답변 1) 

답변 2) 

답변 3) 

|  |
| --- |
| **[Q 8] 매크로 [배점: 10]**  08-macro 라는 디렉토리를 만들고, 그 안에 main.c 소스코드를 아래와 같이 작성하시오.    $gcc -o main-release.exe main.c 명령을 실행하세요. 다음으로, $gcc -DDEBUG -o main-debug.exe main.c 명령을 실행하세요.  **문제 1)** 두개의 exe 파일을 실행하세요. 출력 구문에 어떤 차이가 있나요?  **문제 2)** 이와 같은 차이가 발생하는 이유를 설명하시오.. |

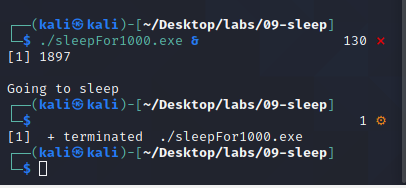
답변 1) main-release.exe 는 “hello world”1줄만 출력이 되고 main-debug.exe 는 3줄이 모두 출력된다

답변 2)DEBUG 가 들어갔기 때문이다

|  |
| --- |
| **[Q 9] Sleep & Kill [배점: 10]**  09-sleep 이라는 디렉토리를 만들고, main.c 함수를 아래와 같이 작성하세요.    소스코드를 컴파일 해서 sleepFor1000.exe 라는 실행파일을 만드세요. 실행파일을 실행하면 다음과 같이 1000초 동안 프로세스가 응답하지 않습니다.    여기서 CTRL+C를 누르면 프로세스를 종료할 수 있지만, 다른 방법으로 종료하는 방법을 찾는 문제입니다. 터미널 프로그램을 하나 더 실행하세요. 아래와 같이 붉은색 터미널에서 sleepFor1000.exe 프로그램이 실행 중이고, 새로 실행한 노란색 터미널이 있습니다.    **문제)** 새로 실행한 터미널(= 노란색 터미널)에서 sleepFor1000.exe 프로그램/프로세스를 종료 시킬 수 있는 방법을 설명하세요. |

답변)

리눅스에 존재하는 kill명령어를 이용하여 강제 종료를 시키면 된다.





**끝! 수고하셨습니다 ☺**

1. 여기서 .exe라는 확장자를 붙인다고 해서 실행파일이 되는 것은 아닙니다. hello.exe 대신, hello.out 등으로 확장자를 아무렇게나 입력해도 실행 파일이 됩니다. 윈도우 사용자에게는 exe 확장자가 실행 파일의 확장자로 익숙하기 때문에 hello.exe 이름을 사용한 것 입니다. [↑](#footnote-ref-1)