CHUNGNAM NATIONAL UNIVERSITY

시스템 프로그래밍

강의 1. 과목 소개

Sep. 2, 2014

http://eslab.cnu.ac.kr

이 과목의 주제

- 컴퓨터 프로그래밍은 추상화(Abstraction)이 대부분
  - 그러나, 본질을 놓치면 안된다
- 컴퓨터 전공의 추상화
  - 자료구조
  - 가상메모리 ?
  - 아이폰의 앱들 – 카카오톡, 페이스북
- 추상화는 한계가 있다
  - 버그가 있을 때
  - 시스템 내부의 동작에 대한 이해가 필요할 때
- 과목의 목표
  - 효율적인 프로그래머를 만들자
  - 컴퓨터하드웨어와 소프트웨어에 대한 본질을 이해
  - 향후 컴퓨터공학 전공 과목들을 위한 준비

충격적인 사실 #1

- *정수가 정수가 아니고, 실수가 실수가 아니다?*
- **Examples**
  - **$x^2 \geq 0$?**
    - Float's: Yes!
    - Int's:
      - 40000 * 40000 --> 1600000000
      - 50000 * 50000 --> ??
  - **(x + y) + z = x + (y + z)?**
    - Unsigned & Signed Int's: Yes!
    - Float's:
      - (1e20 + -1e20) + 3.14 --> 3.14
      - 1e20 + (-1e20 + 3.14) --> ??

컴퓨터 내에서의 연산

- 수학에서는 의미 없는 이상한 값이 갑자기 생기지는 않는다
  - 수식연산은 중요한 수학법칙을 갖는다
- 그러나 컴퓨터에서는 일반적인 수학 법칙을 가정할 수 없다
  - 데이터 표현의 유한성 때문에
- 관찰(Observation)
  - 어떤 추상화가 어떤 상황에 적용된 것인지 이해하는 것이 필요
  - 컴파일러 개발자나 고급 프로그래머들에게는 중요한 주제

충격적인 사실 #2

- *어셈블리어를 알아야 한다 !!!*
- 대개의 경우, 어셈블리 프로그램을 작성할 가능성은 ..... **zero !**
  - 컴파일러가 훨씬 더 우수하고 참을성이 있다
- 어셈블리어를 이해하는 것은 하드웨어 수준의 실행모델을 이해하는데 매우 중요
  - 버그가 있을 때 프로그램의 동작
    - 고차원 언어 모델로는 이해할 수 없다
  - 프로그램의 성능을 튜닝할 때
    - 프로그램의 효율성을 이해하기 위해
  - 시스템 소프트웨어 구현시
    - 컴파일러는 기계어 코드가 목적코드이다
    - 운영체제는 프로세스의 상태를 관리해야 한다
  - **malware를 만들거나 malware와 싸우기 위해**
    - x86 어셈블리를 배운다

어셈블리 코드 예제

```
;*************************************************************************
;Initialize SCCs
;       SCC channels 0, 1, 2, and 3 are initialized. It is assumed
;       that the Peripheral chip selects have been programmed to begin
;       at 0x8000, and the SCCs are PCS0 and PCS1.
;       SPEED: 9600 bits/second from x16 BRG clock from 7.372800 PCLCK
;       CHARS: 8 bits. No parity. 1 Stop bit.
;
;       No interrupts are used.
;*************************************************************************
init_c0:        mov     dx,CH0_CTL              ;address of scc to init
                mov     cx,offset init_c1       ;'return' address
                jmp     init_scc
init_c1:
                mov     dx,CH1_CTL              ;address of scc to init
                mov     cx,offset init_c2       ;'return' address
                jmp     init_scc

init_c2:        mov     dx,CH2_CTL              ;address of scc to init
                mov     cx,offset init_c3
                jmp     init_scc

init_c3:        mov     dx,CH3_CTL
                mov     cx,offset init_muxs
                jmp     init_scc
```

충격적인 사실 #3

- *메모리가 중요하다*

- 메모리크기는 무한하지 않다
  - 메모리는 할당해주고 관리해야 한다
  - 많은 응용프로그램들은 메모리에 영향 받는다
- 메모리 참조 버그는 치명적이다
  - 버그의 결과가 시공간적으로 연관성이 없다
- 메모리 성능은 일정하지 않다
  - 캐시와 가상메모리 효과는 성능에 비선형적 영향을 미친다
  - 프로그램들을 메모리 시스템의 특성에 최적화 시키면 성능이 대폭 개선된다

메모리 참조 버그

```
double fun(int i)
{
  volatile double d[1] = {3.14};
  volatile long int a[2];
  a[i] = 1073741824; /* Possibly out of bounds */
  return d[0];
}
```

```
fun(0)  ->    3.14
fun(1)  ->    3.14
fun(2)  ->    3.1399998664856
fun(3)  ->    2.0000061035156
fun(4)  ->    3.14, then segmentation fault
```

메모리 시스템 성능

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (i = 0; i < 2048; i++)
    for (j = 0; j < 2048; j++)
      dst[i][j] = src[i][j];
}
```

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
  int i,j;
  for (j = 0; j < 2048; j++)
    for (i = 0; i < 2048; i++)
      dst[i][j] = src[i][j];
}
```

59,393,288 clock cycles

1,277,877,876 clock cycles

(Measured on 2GHz Intel Pentium 4)

21.5 **배 더 느리다**!

- 계층적 메모리 구조를 이해하는 사람과 그렇지 못한 사람
- 성능은 메모리 접근 방식에 영향을 받는다
  - 다중 배열에서 어떻게 인덱싱 하는가에 따라 달라진다

충격적인 사실 #4

- *컴퓨터는 프로그램 실행 이외의 여러가지 일을 한다*

- 데이터의 입출력이 필요하다
  - I/O 시스템은 프로그램의 신뢰성과 성능에 매우 중요하다
- 프로그램이 순차적으로 실행되는 것은 아니다
  - 예외적인 순서로(돌발!) 진행될 수 있어야 컴퓨터 다워진다

이 과목의 목표

- 프로그램이 어떻게 하드웨어에서 동작하는지 이해한다.
  - 어셈블리 프로그래밍을 이용해서
- 리눅스의 프로그래밍 환경에 익숙해 지도록 한다
  - 리눅스에서 프로그래밍할 가능성이 높다
- 컴퓨터 시스템을 활용하는 프로그래밍 기법을 학습한다
  - 프로세스의 제어 **Process control**
  - 시그널 **Signal**
  - 메모리 관리 **Memory management**
- 컴퓨터 하드웨어와 소프트웨어의 본질을 이해한다

컴퓨터 시스템에 관한 깊은 이해를 제공한다

강의 일정

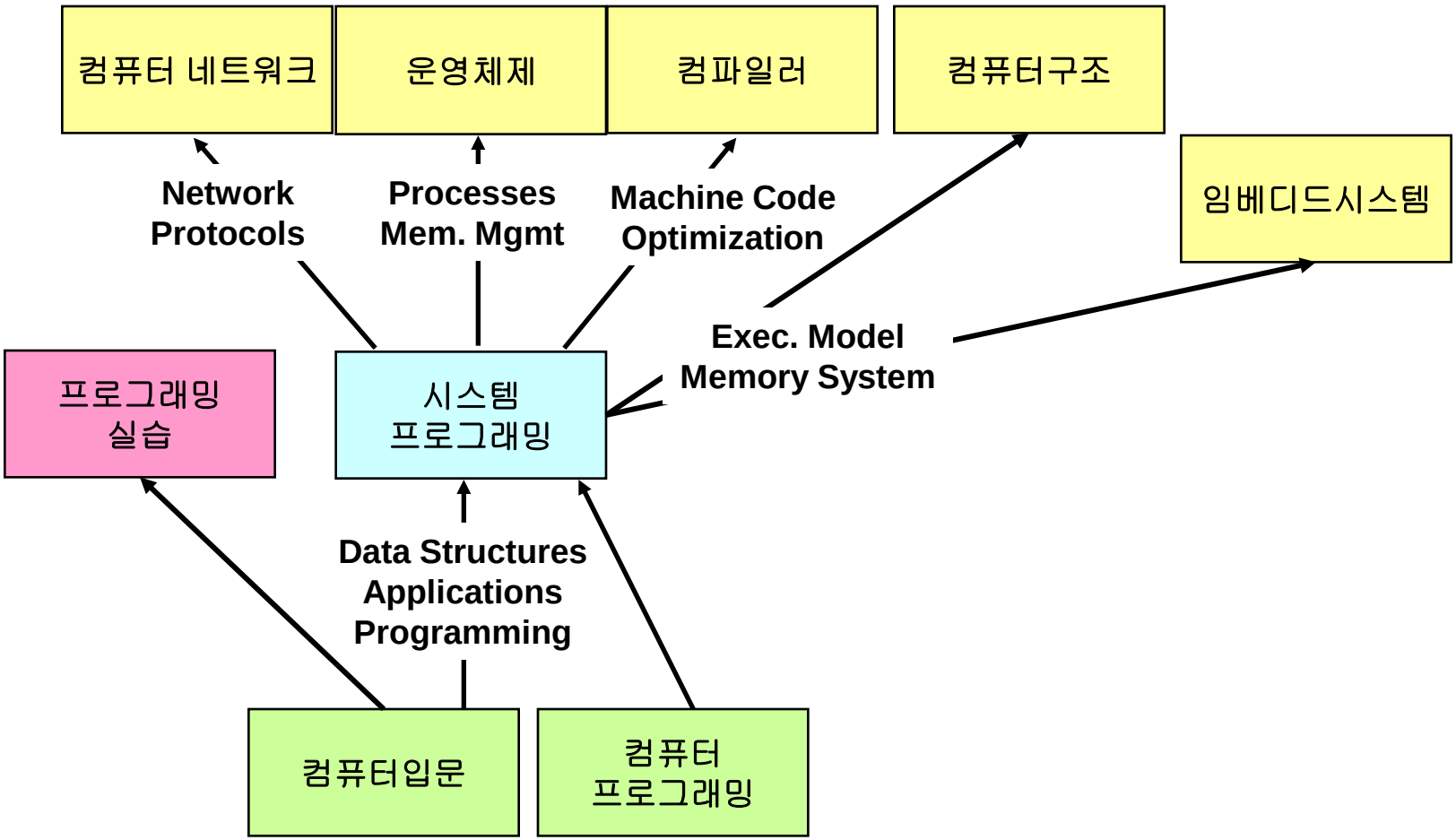
주	날짜	강의실 (화)	날짜	실습실 (목)
1	9월 2일	Intro	9월 4일	리눅스 개발환경 익히기
2	9월 16일	정수	9월 11일	**실수**
3	9월 23일	어셈1 – move	9월 18일	GCC & Make
4	9월 30일	어셈2 – 제어문	9월 25일	Data lab
5	10월 7일	어셈3 – 프로시져 **I**	10월 9일	한글날/휴강
6	10월 14일	어셈3 – 프로시져 II	10월 16일	개천절(10월5일보강) GDB
7	10월 21일	보안	10월 23일	Binary bomb1
8	10월 28일	시험휴강	10월 30일	Binary bomb 2
9	11월 4일	프로세스 1	11월 6일	**Tiny shell 1**
10	11월 11일	**프로세스 2**	11월 13일	Tiny shell 2
11	11월 18일	시그널	11월 20일	Tiny shell 3
12	11월 25일	동적메모리 1	11월 27일	Malloc lab1
13	12월 2일	동적메모리 2	12월 4일	Malloc lab2
14	12월 9일	**기말고사**	12월 11일	Malloc lab3
15	12월 16일	**Wrap-up/종강**		

이 수업이 끝날 때 여러분은,

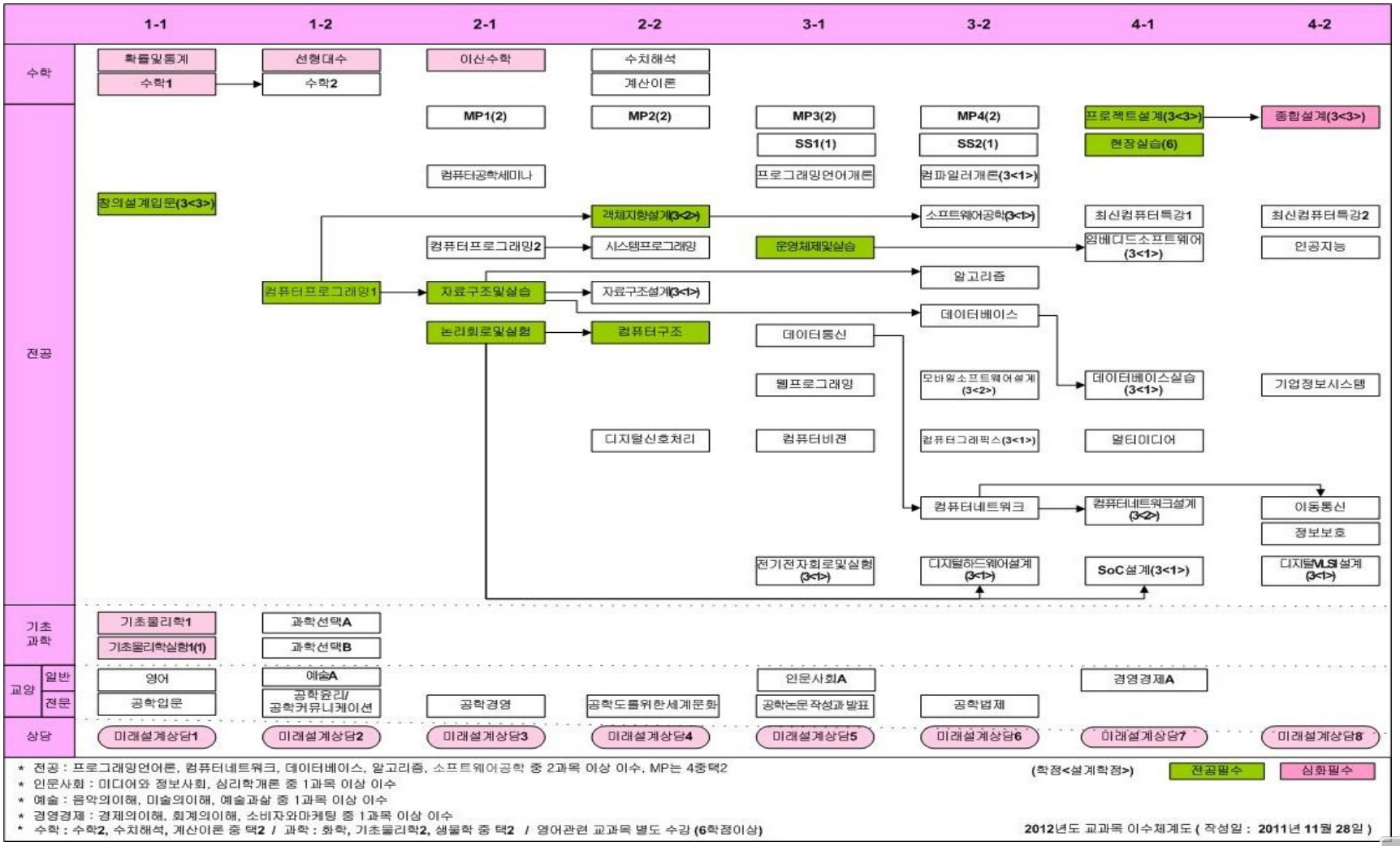
- 리눅스에서 C 프로그래밍 하는 기술을 습득한다
- IA-32 어셈블리어를 사용할 수 있게 된다.
- 다른 프로세서의 어셈블리어를 쉽게 습득할 수 있게 된다.
- 저 차원의 디버깅 기술을 사용할 수 있게 된다.
- 컴퓨터 내부에서의 숫자의 표현방법을 이해하게 된다
- 고급언어가 어떻게 번역되고, 컴퓨터에서 처리되는지 이해하게 된다.
- 운영체제의 예외처리 과정을 이해한다
- 프로세스를 제어하는 프로그램을 작성할 수 있다
- 운영체제의 메모리 관리 원리를 이해하게 된다

막강한 준-프로 시스템 프로그래머가 된다!

컴퓨터 전공과목과의 결정적 관계



교과목 이수체계도(2012학번)



담당교수

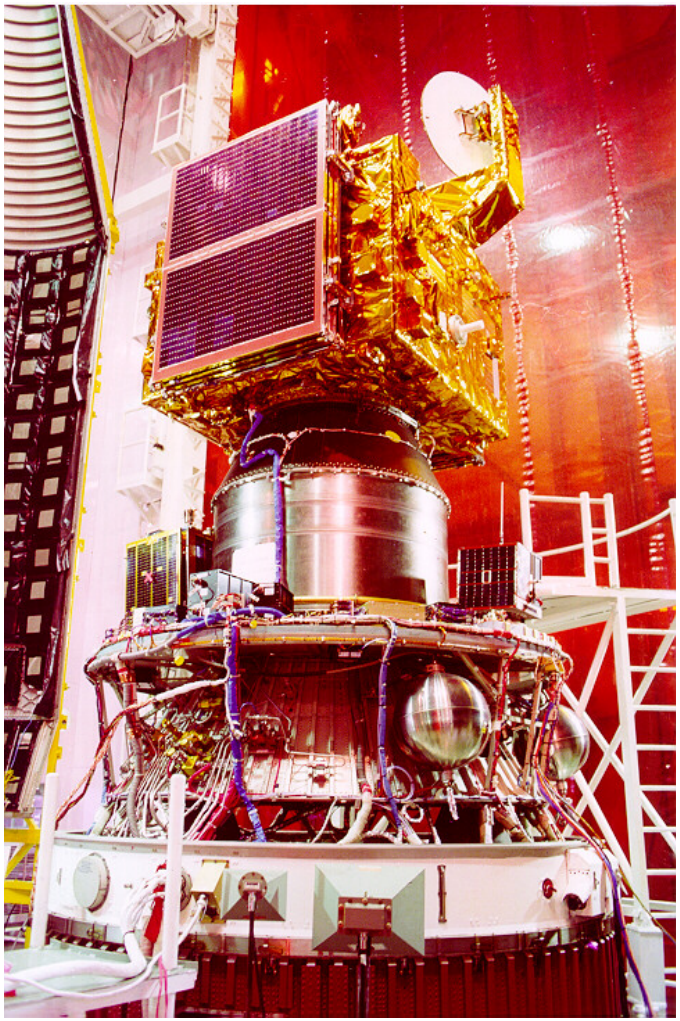
Hyungshin Kim, Professor of CSE, CNU

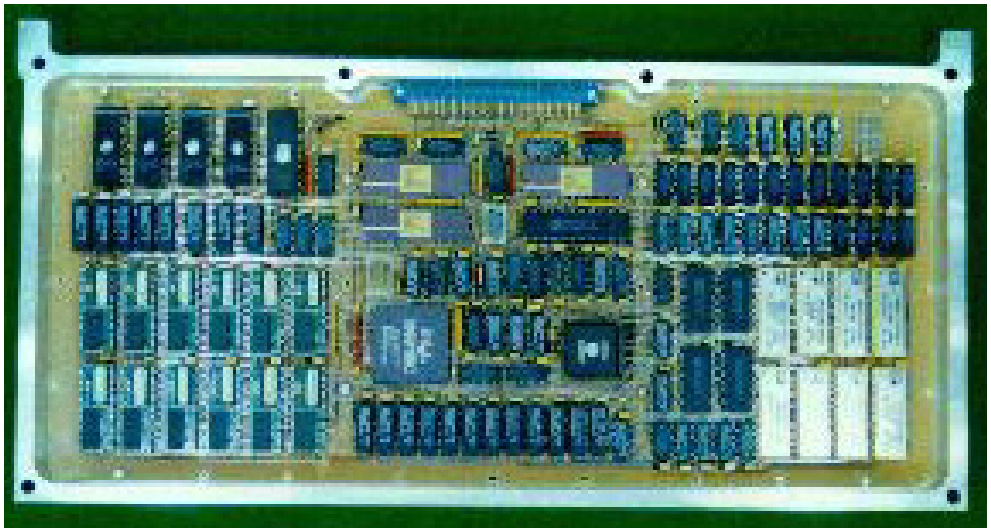
BS	Computer Science at KAIST
MS	Satellite Communication Engineering, Univ. of Surrey, UK
PhD	Computer Science at KAIST
90-92	Researcher, SSC, UK
92-01	Researcher, SaTReC, KAIST
03-04	Post-Doc, Carnegie Mellon University, USA

Office : E5 521
Voice : 821-5446
E-mail : hyungshin@cnu.ac.kr
Web : http://eslab.cnu.ac.kr
Lab : Embedded System Lab. (ESL), E5 533
Office hour : TBD

아리안과 나

나와 우주





National Space Lab
NSL

과목 개요 – 1 / 2

- 조교 : 박시형(02반), 한동건(03반)
- 교재 :
  - Computer systems : A programmer's perspective, 2판, by Randal E. Bryant and David O'Hallaron, Prentice Hall
- 참고문헌

- 교과목 홈페이지
  - http://eslab.cnu.ac.kr, lecture menu, 2014 Undergraduate
- 선수지식
  - 능수 능란한 C 언어 프로그래밍

과목 개요 – 2 / 2

- 성적평가
  - 중간고사 : 20%
  - 기말고사 : 30%
  - 실습 : 30%
  - 출석 : 10%
  - 숙제 : 10%
  - 주의 ! 2학년이 아닌 고학년의 수강 시 고학년의 수강수준을 감안하여 높은 기준으로 별도로 평점 부여
- 출석
  - 2회까지는 눈 감아줌. 3회 결석부터 2점씩 감점
  - 실습포함 10회이상 결석시 F(학칙에 의거)
- 숙제
  - 예습 교재읽기, 문제풀이, 프로그래밍 등 다양한 유형
  - 마감일을 넘기는 경우 50% 감점
- 숙제/실습 복사에 대해
  - 부도덕한 행위에 대해서 F 부과(실습 숙제 카피 주의!!!)
  - 주요 실습 평가시 구두 평가 실시 – 구두 평가 결과로 실습의 진정성 평가

이 과목의 역사

- 2005 가을학기 Unix programming 2-1-2 : 4.1/5.0
- 2006 가을학기 Unix system programming, 3-2-2 : 3.9/5.0
  - 2005 Unix programming + System programming
  - 교재 변경=> Programmer's perspective …
  - REB 책 후반부로 구성 (Ch.8 – Ch.13)
  - 영어 강의 1개반, 일반강의 1개 반
- 2007 가을학기 – Unix 프로그래밍 : 4.1/5.0
- 2009 가을학기 – Unix 프로그래밍 : 4.2/5.0
- 2010 가을학기 – 시스템 프로그래밍 : 4.2/5.0
  - 어셈블리 프로그래밍 + Unix 프로그래밍
- 2011 가을학기 4.0/5.0, 3.9/5.0
  - 전공필수로 전환
- 2012 가을학기 3.52/5.0, 3.89/5.0
- 2013 전공선택으로 복귀 4.127/5.0, 4.341/5.0

2014 시스템 프로그래밍 강의 구성

- 25분 강의 3분 휴식, 20분강의
- 난이도 높은 실습에 대한 지도 강화
- 도전적인 실습과제의 구성
- C 프로그래밍 능력 강화
- 리눅스에서 프로그래밍 도구 활용 능력 강화
- 핵심 실습의 카피 여부 집중 검증
- 2학년 수준에 적합한 강의내용
- 컴퓨터 프로그래밍 본질의 이해
- 영어 원서 강의 교재 적응
- 어리버리한 2학년들의 전공으로의 연착륙

다음주를 위한 준비

- 새 학기를 맞는 여러분의 결심
  - 강의는 열심히, 생활은 즐겁게
  - 전공으로 진입하는 중요한 기회의 학기
  - 적절한 수준의 수강신청
- 예습 숙제 1.
  - 교재 **2.1.4 Addressing and Byte Ordering(pp.73-80)** 읽고 **A4 1**쪽 이내로 요약
  - 다음주 수업시간에 제출
- 이번주 실습 1
- 홈페이지 가서 강의자료 내려받기.