본격적인 작업에 들어가기에 앞서 컴파일을 위한 설정을 해야 한다.

우리의 확장모듈을 컴파일할때에 /usr/lib에 설치된 libformat.so 와 함께 링크를 해야 하기 때문에 config.m4의 한줄을

추가된 config.m4의 내용은 다음과 같다. (기본 주석문 제거)

```
if test "$PHP_LIBFORMAT" != "no"; then

  dnl dnl로 시작하는 문장은 주석문입니다.
  dnl /usr/lib/libformat.so와 함께 컴파일 해야 하기 때문에 아래와 같이 적습니다.
  dnl libformat.so에서 lib는 기본적인 이름이기 때문에 format 만 적습니다.

  PHP_ADD_LIBRARY_WITH_PATH(format, /usr/lib, LIBFORMAT_SHARED_LIBADD)
  PHP_EXTENSION(libformat, $ext_shared)
fi
```

자 ext_skel에 의해 만들어진 libformat.c 를 편집해 보자.

vi ext/libformat/libformat.c

ext_skel은 vim의 SourceCode folding 주석을 달아주기 때문에 이런 형태로 소스코드가 보일수 있다.

+-- 11 lines: libformat_functions[]----------

이때 당황하지 말고 + 에 커서를 데고 l이나 -> 키를 누르면 함수나, 긴 선언문이 쫘악 펼쳐진다.

만약 vim의 folding 효과가 거부감을 일으킨다면. 맨 아랫줄의 이 주석을 지우고 다시 열면 된다.

```
/*
* Local variables:
* tab-width: 4
* c-basic-offset: 4
* End:
* vim600: sw=4 ts=4 tw=78 fdm=marker
* vim<600: sw=4 ts=4 tw=78
*/
```

소스 코드의 libformat_functions[]를 주목할 필요가 있다.

ext_skel이 만들어준 테스트용 php code에 get_extension_funcs($module) 함수가 있었던 것을 기억하는가?

PHP의 확장모듈은 소속된 함수들을 libformat_functions라는 변수에 정해 놓는다.

최초에 libformat_functions는 아래와같이 선언되어 있다.

```
 function_entry libformat_functions[] = {
    PHP_FE(confirm_libformat_compiled,  NULL)       /* For testing, remove later. */
    {NULL, NULL, NULL}  /* Must be the last line in libformat_functions[] */
};
```

우리가 만들 함수는 format_string 와 format_file 이다. 이 두개의 함수를 첨가해보자.

```
function_entry libformat_functions[] = {
    PHP_FE(confirm_libformat_compiled,  NULL)       /* For testing, remove later. */
    PHP_FE(format_string,   NULL)       /* format_string */
    PHP_FE(format_file, NULL)       /* format_file */
    {NULL, NULL, NULL}  /* Must be the last line in libformat_functions[] */
```

```
};
```

get_extension_funcs($module)는 libformat_functions 에 저장된 함수정보를 {NULL,NULL,NULL}이 나올때 까지 차례
Python의 C확장도 이것과 흡사하다. 나중에 기회가 되면 Python과 PHP의 C확장에 대해 비교해 보겠다.
PHP_FE는 미리 정의된 매크로이다.

자 이제 실제 format_string와 format_file 함수 작성에 앞서 confirm_libformat_compiled 함수를 살펴보자.
이 함수는 ext_skel이 예제용으로 만들어놓은 함수이기 때문에 테스트후 지우기 바란다.

```
PHP_FUNCTION(confirm_libformat_compiled)
{
    char *arg = NULL;
    int arg_len, len;
    char string[256];

    // 함수 아규먼트의 개수를 센다.
    //  zend_parse_parameters함수는 넘어온 아규먼트로 부터 "s", 즉 스트링형식의 아규먼트 하나를 arg에 참조
    //  arg_len에는 넘어온 스트링의 길이가 저장된다.
    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "s", &arg, &arg_len) == FAILURE) {
        return;
    }

    // 아래와 같은 문자열을 만들고 리턴한다.
    len = sprintf(string, "Congratulations! You have successfully modified ext/%.78s/config.m4. Module %.78
    RETURN_STRINGL(string, len, 1);
}
```

PHP가 함수의 아규먼트를 처리하는 모습을 보면 재미있는 것을 발견할 수 있다.
저데로라면 function Foo($bar) 로 정의된 함수를 호출할때에 Foo("bar")로 호출하건 Foo("bar","bar2","bar3") 로 호출
아규먼트 하나만 처리하기 때문이다.
실제로 함수관련 에러에서 아규먼트가 많다고 나는 에러는 못보았을 것이다.

PHP의 아규먼트 처리와 리턴에 관한 자세한 내용은 php manual의 "V. Extending PHP 4.0"에 잘 정리되어 있다. 여기서

이제 우리가 만드려는 함수 format_string와 format_file에 대해 알아보자. 이 함수들은 libformat를 컴파일하고 설치했을
일들이다.

```
  /* libformat 라이브러리가 제공하는 랭귀지인지 판별 */
  int format_valid_lang(const char *language);

  /* 이 함수들을 통해 리턴되는 스트링은 dynamically allocate된 메모리 */
  /* 참조이기 때문에 반드시 메모리 free 해 주어야 한다. */
  char *format_fd     ( int fd, const char *language );
  char *format_fp     ( FILE *fp, const char *language );
  char *format_file   ( const char *filename, const char *language );
  char *format_string ( const char *str, const char *language );
```

libformat C library가 제공하는 함수는 이게 전부다. 정말 심플하기 때문에 쉽사리 확장모듈 만들생각이 들었고, 또한 이
각함수들의 두번째 아규먼트인 language는 현재 "c", "c++", "java", "python", "verilog", "vhdl" 만 처리할 수 있다. php
PHP에서 fd나 파일포인터를 사용할수 없기 때문에 우리는 아래 두개의 펑션만 구현한다.
이 두개의 구현으로도 충분한 활용이 가능하다.

format_file는 파일명과 언어이름을 입력하면 Highlight Syntax 된 HTML을 리턴하고, format_string는 소스코드를 직접

자 아까 보았던 confirm_libformat_compiled를 응용하여 만들어 보자.

먼저 ext/libformat/libformat.c 의 상단부에 헤더파일 인클루드를 적자.

```
#include "format.h"
```

하단부에 아래 코드를 추가하자. PHP확장모듈작성시에 대부분 매크로를 이용하므로 일반적인 C코드와는 좀 달라보인다

```
PHP_FUNCTION(format_string)
{
    char *str = NULL;
    int str_len;
    char *language = NULL;
    int language_len;
    char *output;

    // 스트링 방식의 아규먼트 2개를 받는다.
    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "ss", &str, &str_len, &language, &language_len) ==
        return;
    }

    // 두번째 파라메터인 language가 지원가능한 언어인지 검사한다.
    if(format_valid_lang(language))
    {
        // libformat C library의 함수를 직접 사용하여  output에 참조시킨다.
        output = format_string(str, language);
        if(output)
        {
            // output 리턴후에 반드시 메모리 free를 해주어야 하지만 PHP는 이작업을 대신해준다.
            RETURN_STRING(output, 1);
        }
        return ;
    }
    else
        return;
}
```

```
PHP_FUNCTION(format_file)
{
    char *str = NULL;
    int str_len;
    char *language = NULL;
    int language_len;
    char *output;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "ss", &str, &str_len, &language, &language_len) ==
        return;
    }
    if(format_valid_lang(language))
    {
        output = format_file(str, language);
        if(output)
        {
            RETURN_STRING(output, 1);
        }
        return ;

    }
    else
        return;
}
```

ext/libformat/php_libformat.h를 열어 format_string와 format_file을 선언하자.

```
PHP_FUNCTION(format_string);    /* format_string */
```

```
PHP_FUNCTION(format_file);  /* format_string */
```

자 이제 make 하고 테스트 해보자

```
<?
print format_string("import time","python");
?>
```

이런식으로 필요한 확장모듈을 만들어 보고, PHP의 구조에 대해 고찰해보자.

확장모듈 구조가 있는 PHP는 정말 못할것이 없어보인다.

특히 php-gtk라는 프로젝트는 PHP확장모듈을 이용한 GUI 인터페이스 작성을 가능하게 해준다.