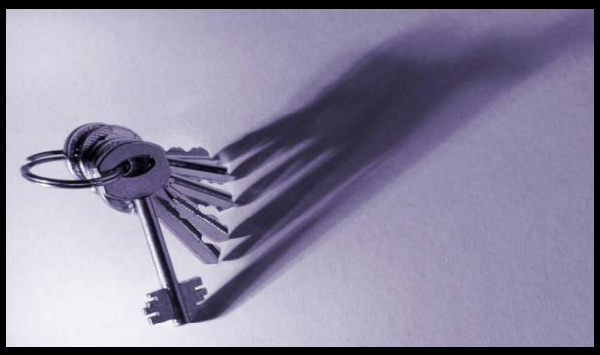
PRO*C PRECOMPILER의 이해와 활용



Getting the most out of MetaLink

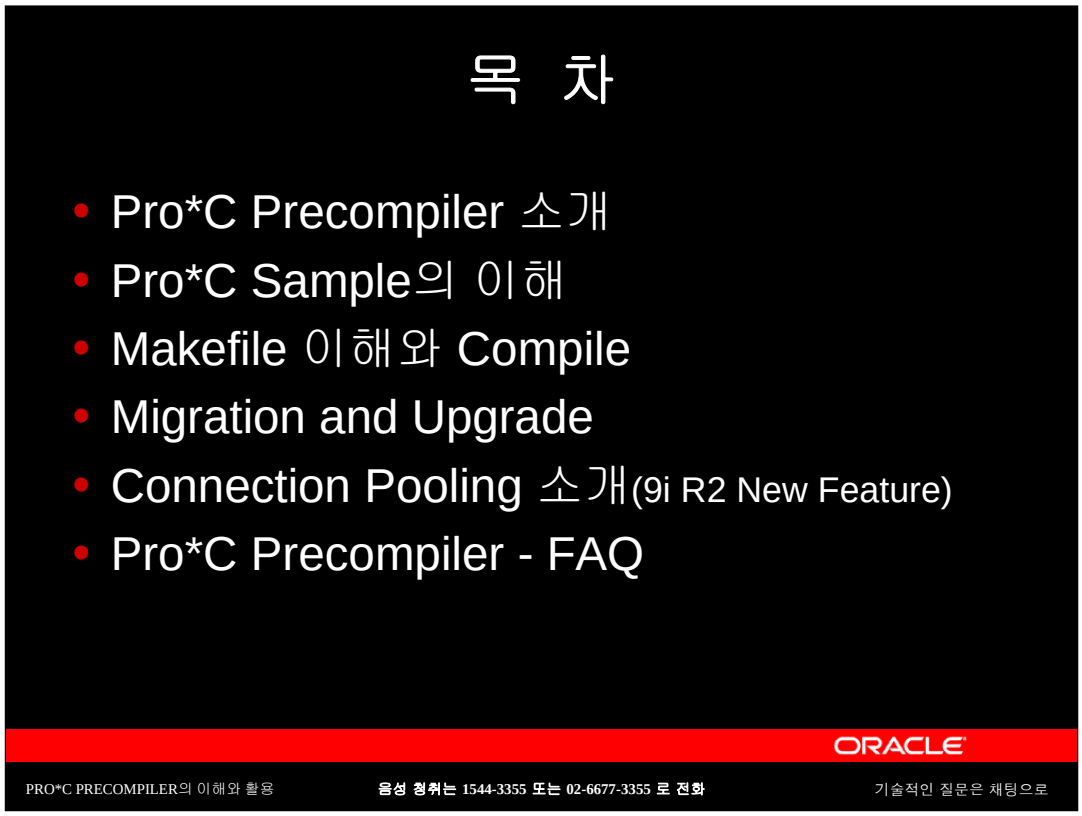
손두근

한국 오라클 (주) 제품지원실

ORACLE

오늘 세미나에서는 Oracle Precompiler중에서 Pro*C에 대해 설명 드리고자 합니다.

“Pro*C Precompiler의 이해와 활용” 세미나를 시작 하겠습니다.



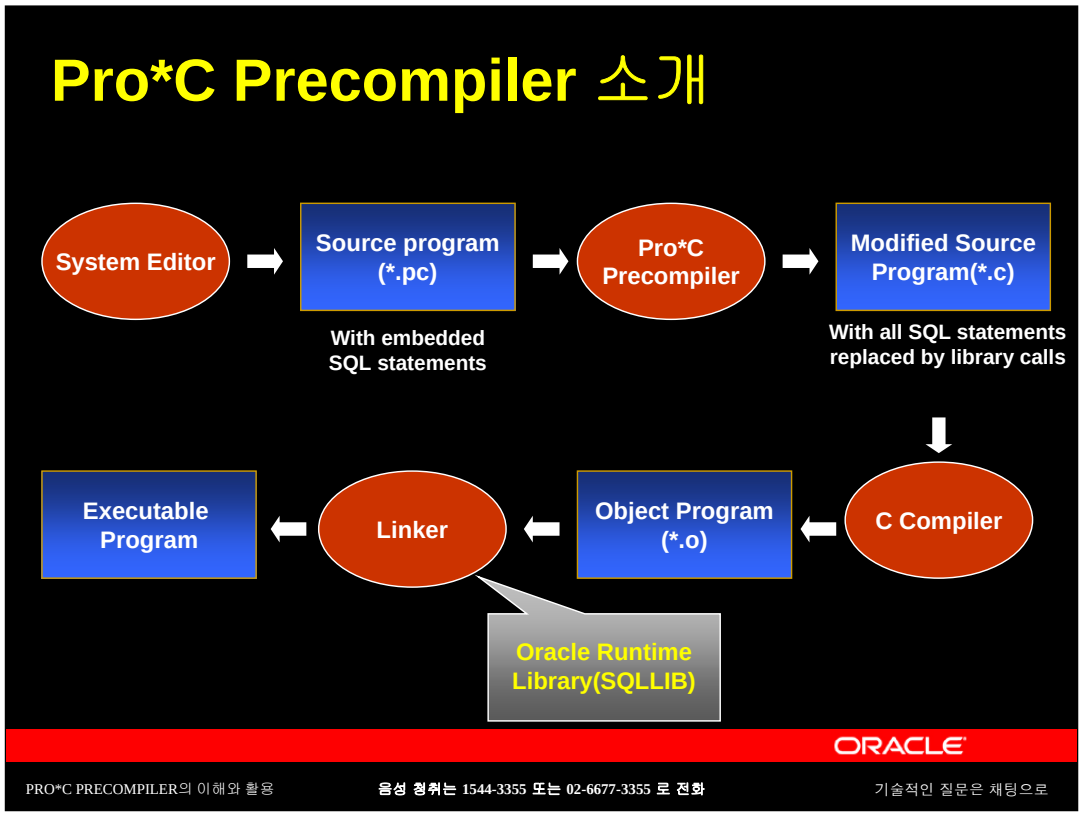
본 세미나에서 진행할 내용은 다음과 같습니다.

먼저, Pro*C가 어떻게 처리되고, 어떻게 install하며 무슨 options들이 있는지 알아보겠습니다.

다음으로, Pro*C install후에 설치되는 sample code와 makefile에 대해 설명 드리고, 이를 이용하여 어떻게 compile하는지 알아보겠습니다.

또한, DB server의 upgrade와 migration시에 pro*c program에 대한 조치방법과 Oracle v 9.2.x version의 새로운 기능인 connection pooling에 대해 알아보겠습니다.

마지막으로, Pro*C Precompiler에 대해 많이 질문하시는 내용을 정리하였습니다.

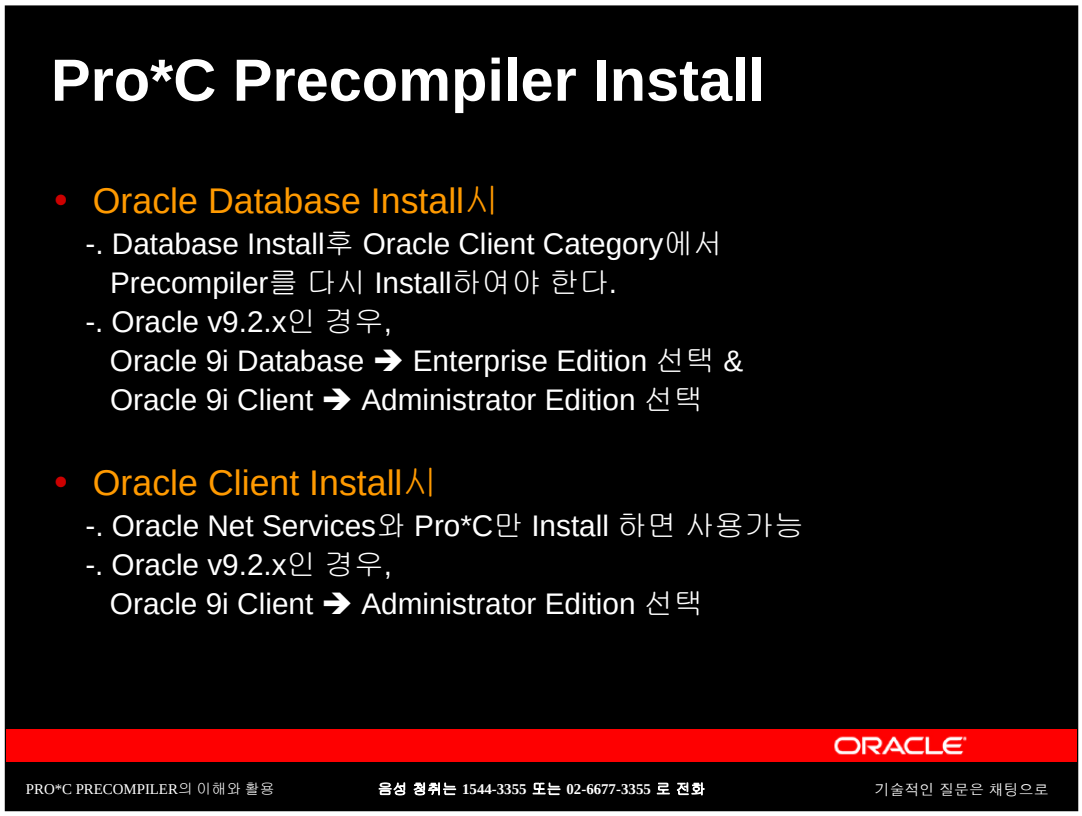


Pro*C Precompiler 소개

사용자는 Embedded SQL문을 포함하는 source program을 작성하며, 이때 파일 확장자는 *.pc가 됩니다.

이를 Pro*C precompiler를 통해 Embedded SQL문을 library call로 대체하여 확장자가 *.c인 c program을 생성 합니다.

여기서 생성된 C program을 C compiler를 통해 object program을 생성하고, oracle runtime library를 link하여 실행 화일을 생성하게 됩니다. 이런 일련의 과정을 makefile에 기술하여 간단히 compile할 수 있습니다.

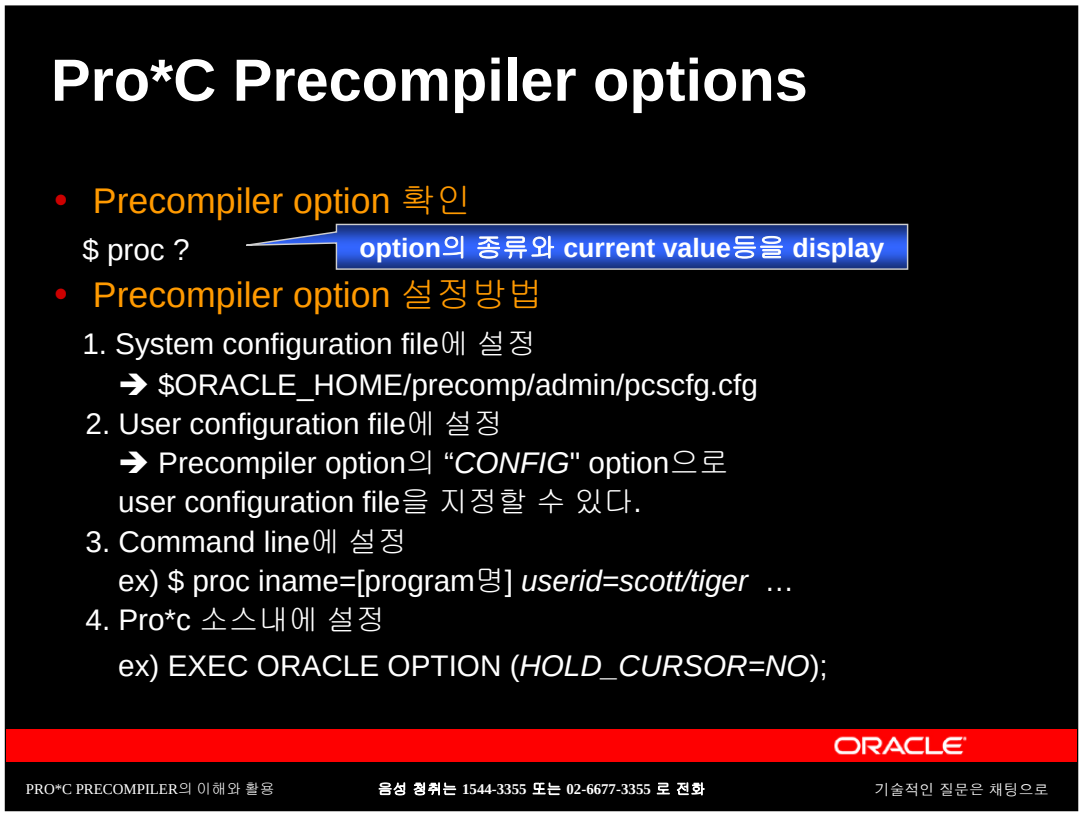


Pro*C Precompiler Install

Oracle 8i 이후, Pro*C Precompiler는 oracle database설치 시에 default로 설치되지 않습니다.
그러므로, oracle database install후 oracle installer를 다시 기동하여 Pro*C Precompiler를 한번 더 install하여야 합니다.

Oracle v 9.2.x에서 Pro*C 를 사용하기 위해서는 Oracle 9i database -> Enterprise edition를 선택하여 DB Server를 install한 후, 다시 Oracle 9i client -> Administrator edition을 선택하여 Pro*C Precompiler를 install하여야 한다.

Pro*C precompiler를 Oracle database와는 별개로 client에서도 이용이 가능하며, 이때 Oracle Net Services(또는 NET8, Oracle 8i 이전의 경우 SQL*NET 제품)와 Pro*C만 install하면 사용이 가능합니다.
Oracle v 9.2.x에서 Oracle 9i Client -> Administrator edition을 선택하여 Install하면 Oracle Net Services와 Pro*C가 install된다.



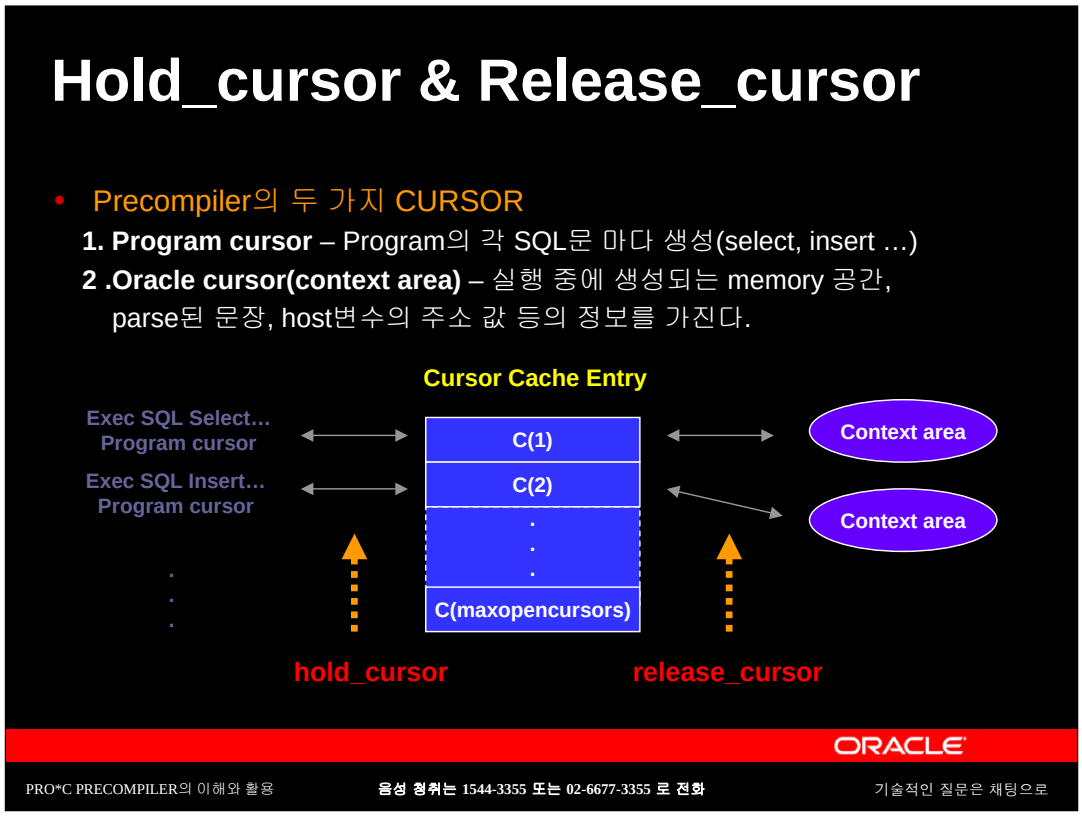
Pro*C Precompiler option

Pro*C에 대한 option, current value 및 설정 가능한 value 값 등의 간단한 설명을 확인하려면,
$ proc ?
하시면 됩니다.

Precompiler option을 설정하는 방법은 4가지가 있습니다.

1. System configuration file에 설정합니다. $ORACLE_HOME/precomp/admin/pcscfg.cfg 파일에 한 line씩 설정하고, space없이 기술하여야 합니다. 이 곳에 설정하면 모든 pro*c program에 영향을 미칩니다.
2. User configuration file에 설정합니다. Precompiler option중에 "CONFIG" option으로 user configuration file을 지정하고, 이 파일에 system configuration file과 같은 방법으로 option들을 설정합니다.
3. Precompile하는 command line에 설정합니다. 특정 program에만 option을 설정하는 경우 사용합니다. Ex) proc iname=sample1 userid=scott/tiger sqlcheck=semantcis
4. Pro*C 소스내에 "exec oracle option" embedded SQL문으로 설정할 수 있습니다. 이 후에 문장에 option이 적용됩니다.

** Precompiler option종류에 따라서 설정하는 방법에 제약이 있습니다.
예를 들면, "code", "mode"같은 option들은 pro*c 소스 내에서 설정할 수 없습니다.



Hold_cursor 와 Release_cursor

Pro*C option중에는 hold_cursor와 release_cursor가 있습니다.

우선 Precompiler에서 사용하는 두가지 cursor에 대해 알아보겠다.

1. Program cursor – SQL문으로 인해 생기는 data 구조로, Program의 각 SQL문 마다 생성된다. Select문 뿐만 아니라, Insert문도 하나의 cursor가 생성된다.
2. Oracle cursor (context area 라고 함.) – 실행 중에 생성되는 memory 공간으로, parse된 문장과, host변수의 주소 값, 그 외에 SQL문을 실행하기 위한 필요한 정보를 가지고 있다.

두 개의 cursor와 cursor cache entry사이의 관계를 설정하는 것이 hold_cursor와 release_cursor option이며, 여기서 cursor cache entry는 maxopencursors size에 의해 할당되는 cursor cache이다.

- HOLD_CURSOR

Pro*c program과 cursor cache entry사이에 cursor 재사용여부를 설정한다.

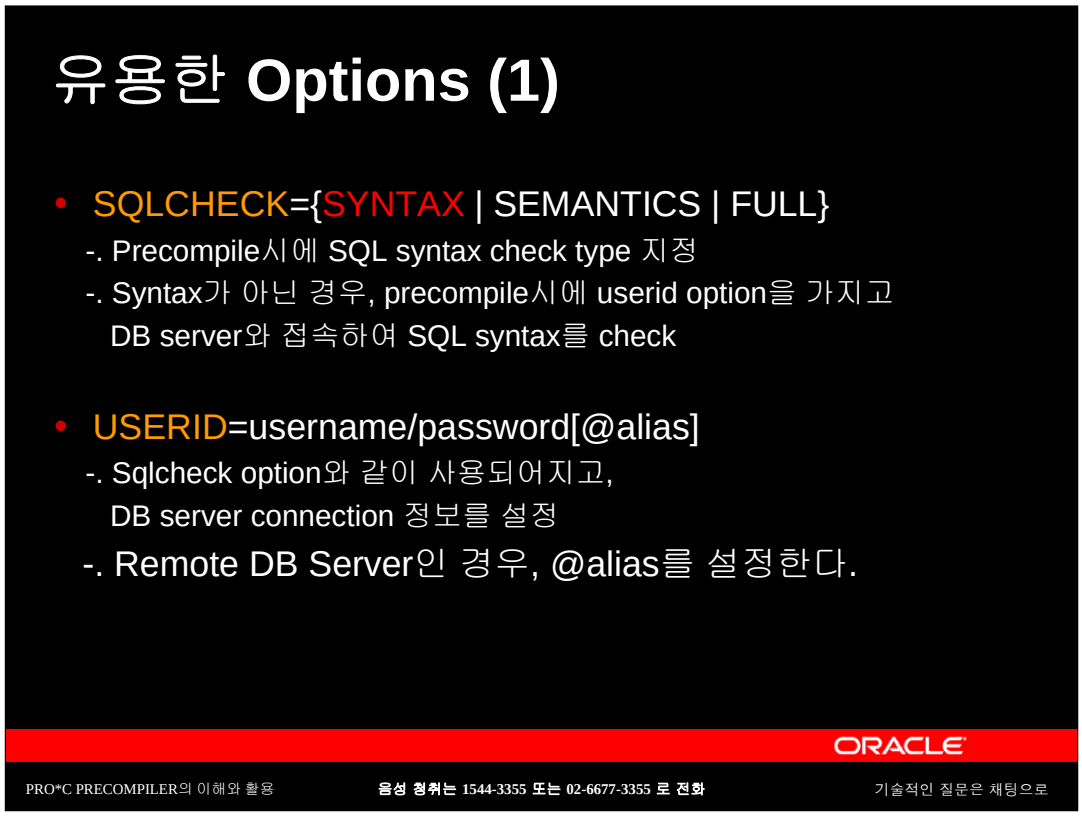
hold_cursor=yes인 경우에는 cursor cache에서 cursor를 재사용하지 못하게 한다.

- RELEASE_CURSOR

cursor cache entry와 context area내에 oracle cursor사이의 관계를 정의하며, 이 option은 parse된 문장이 실행한 이후의 상태를 관리한다.

“release_cursor=no”인 경우에는 context area에 oracle cursor를 계속 유지시켜 주며, 이때 할당된 memory는 다시 사용하기 위해 계속 유지 되어진다. 이 때 사용된 memory를 바로 풀어주기 위해서는 “release_cursor=yes”를 사용하여야 합니다.

Pro*C에서 Cursor를 재사용하기 위해서는 hold_cursor=yes, release_cursor=no 여야 한다.



유용한 option

다음으로, 여러가지 Pro*C option중에서 자주 사용되는 option들에 대해서 설명드리겠습니다.

• SQLCHECK option

Precompile시에 SQL syntax check type을 지정하며, syntax, semantics, full로 설정할 수 있고, default는 syntax이다.

semantics는 full과 같은 값으로 precompile시에 “userid” option으로 DB server와 접속하여 embedded SQL문을 check한다.

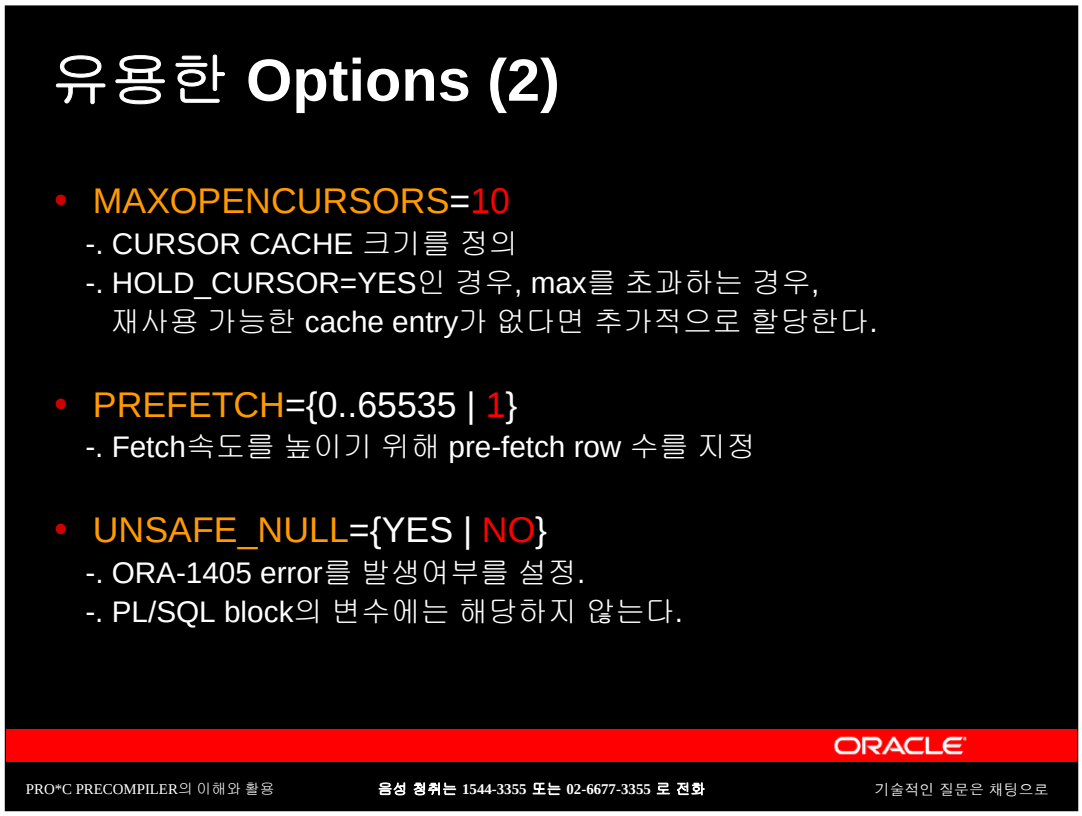
Pro*C소스에 PL/SQL Block이 포함되어 있다면, sqlcheck option을 syntax가 아닌 semantics나 full로 설정하여야 한다.

• USERID option

“sqlcheck” option과 같이 사용되어지며, DB server connection정보를 설정한다.

remote DB Server인 경우, userid/password@alias로 설정한다.

“sqlcheck” option과 마찬가지로 Pro*C소스에 PL/SQL block이 있는 경우, 반드시 설정하여야 한다.



- MAXOPENCURSORS option

앞에서 설명드린 hold_cursor와 release_cursor사이에 사용되는 cursor cache 크기를 정의합니다.

만약, HOLD_CURSOR option을 YES로 설정하고, cursor의 개수가 maxopencursors 크기에 도달하였다면, 실행 시에 cache entry를 동적으로 할당하게 됩니다. 이때, DB Server의 open_cursors 크기만큼 할당될 수 있습니다.

- PREFETCH option

Select의 fetch 속도를 높이기 위해 pre-fetch row 수를 지정합니다.

0에서 65535까지 설정이 가능하고, 0인 경우, prefetch기능을 사용하지 않는다.

default가 1이므로, fetch속도를 향상시키기 위해서는 이 option을 tuning하여야 합니다.

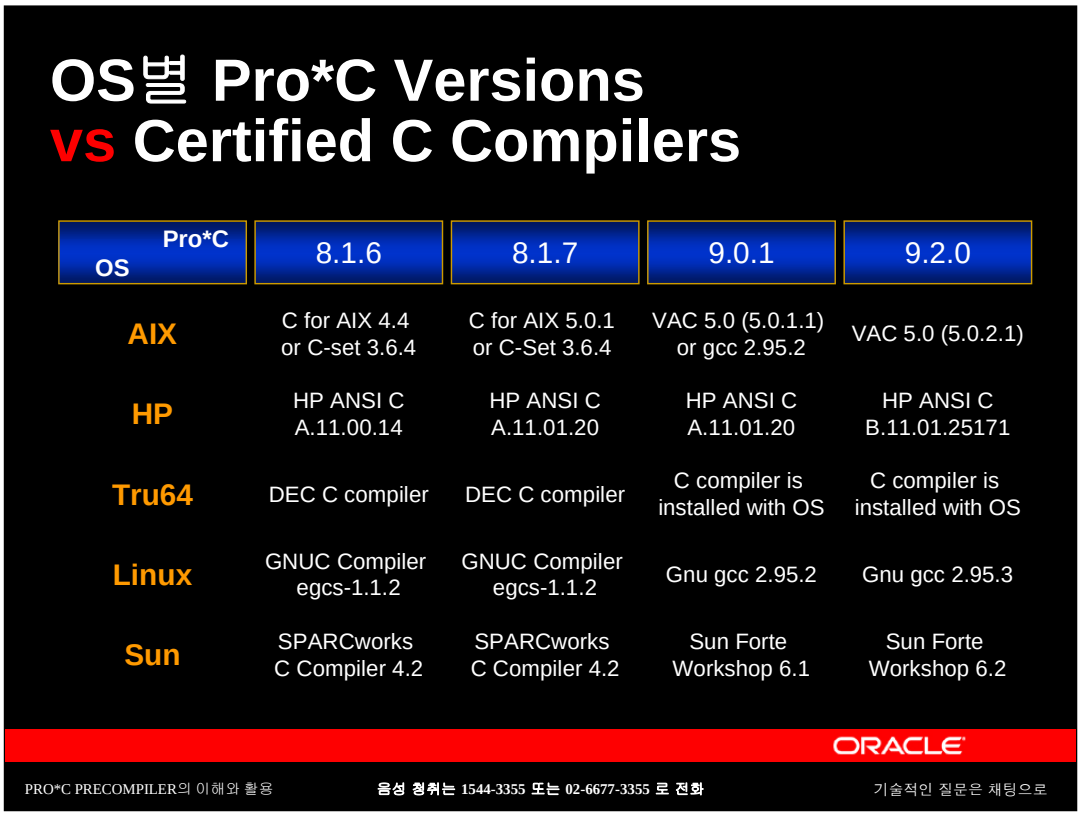
Oraccle8i부터 지원가능한 option입니다.

- UNSAFE_NULL option

pro*c에서 fetch된 값에 null인 경우, ora-1405 error가 발생하는데, unsafe_null=yes인 경우, ora-1405 error가 발생하지 않는다.

PLI/SQL block의 host variable에는 해당되지 않는다.

만일 precompile시 이 option을 사용하지 않을 경우, null값을 처리하기 위해서는 Pro*C Pregram에서 indicator 변수를 사용하여야 한다.



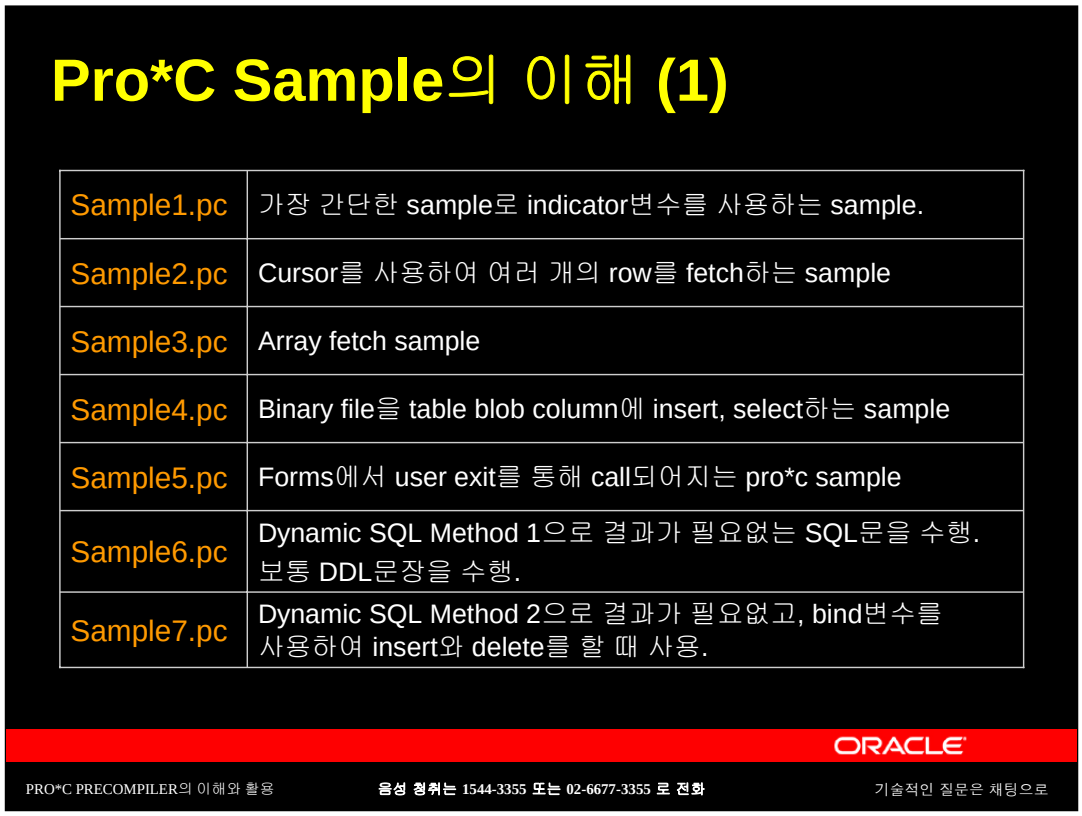
OS별 Pro*C version과 certified된 c compiler version.

OS별로 Pro*C와 C compiler의 상호관계를 알아보겠습니다.

Pro*C를 사용하기 위해서는 각 OS별로 상호 연동이 가능한 C compiler가 존재하여야 합니다.
Pro*C Precompiler 8.1.6, 8.1.7, 9.0.1, 9.2.0에 대해 간단히 정리하였습니다.

특히, SUN Solaris와 Linux인 경우, 위에서 정의된 compiler가 아니거나, 다른 version을 사용하는 경우, 실행 시에 core가 발생하거나, compile자체가 안 되는 경우가 발생합니다.

보다 자세한 내용은 각 H/W Platform에 맞는 DB Server의 “Installation guide”를 참고하면 해당 OS에서 제공하는 연동 가능한 Compiler의 버전을 확인할 수 있습니다.



Pro*C sample의 이해

Pro*C sample에 대해 알아보겠습니다.

Pro*C를 Install하면, $ORACLE_HOME/precomp/demo/proc 에 Pro*C sample이 Install됩니다.
이 sample들을 잘 이해하면, Pro*C Program작성시에 아주 유용하게 사용될 수 있습니다.
그리고, Pro*C Precompiler에서 지원되는 여러가지 기능을 쉽게 확인하실 수 있습니다.

Pro*C sample은 sample1에서 sample12과, cpdemo1, cpdemo2로 구성되어 있습니다.
이중에서 자주 이용되는 몇가지 sample에 대해 설명 드리겠습니다.

Sample1 : 가장 간단한 sample로 Pro*C에서 NULL값을 handling하기 위해 indicator변수를
사용하는 sample입니다.
unsafe_null option을 이용하여 NULL값을 fetch할 수는 있지만,
NULL값을 insert하거나, update하는 경우, indicator변수를 사용하여야 합니다.

Sample3 : select시에 array 변수를 사용하여 한번에 여러 개의 row를 fetch하는 sample입니다.
물론, insert, update문도 array 변수를 이용하여 여러 개의 row를 처리할 수 있습니다.

Sample4 : blob type column에 binary file을 insert하거나, select하는 sample입니다.
DB Version에 따라서 blob type이나, long raw type을 이용합니다.

Pro*C Sample의 이해 (2)

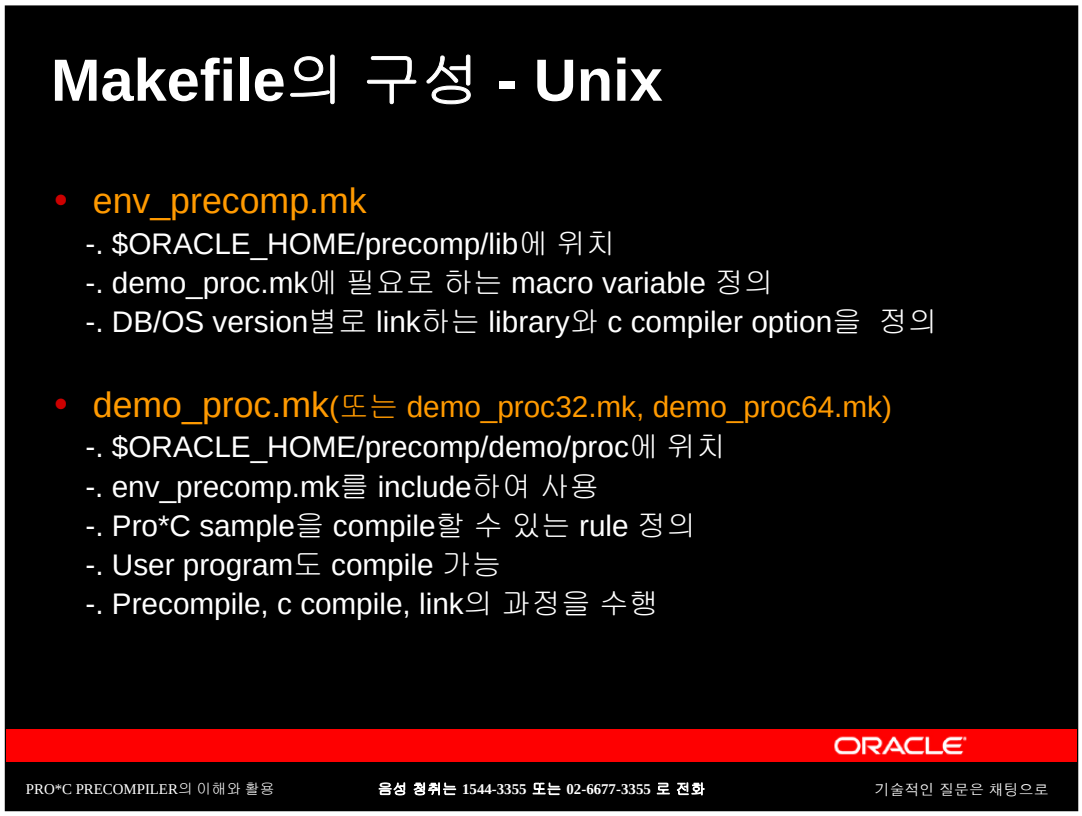
Sample8.pc	Dynamic SQL Method 3으로 결과를 fetch하는 select문 수행
Sample9.pc	Pro*C에서 stored procedure를 call하는 sample
Sample10.pc	Dynamic SQL Method 4으로 SQL문, select list, 조건 등이 모두 가변적일때 사용. 즉, SQL*Plus와 같은 형태.
Sample11.pc	Cursor변수 사용하는 sample로 stored procedure에서 cursor변수를 받아서, pro*c에서 fetch하는 sample.
Sample12.pc	Dynamic SQL Method 4를 이용한 array fetch하는 sample
Cpdemo1.pc	Multithreaded Applications 9.2.X Only
Cpdemo2.pc	Multithreaded Applications에 connection pooling을 사용

ORACLE

Sample10 : Dynamic SQL Method 4을 이용하여 작성된 sample로,
SQL문에서 select list, 조건, table등이 모두 가변적인 경우 사용합니다.
Compile후 실행하면, SQL*Plus와 같은 형태가 됩니다.

Sample12 : sample10과 같은 Dynamic SQL Method 4을 이용하여 작성된 sample로
select문에 대해서 array변수를 이용하여 fetch하는 sample입니다.

Cpdemo1, Cpdemo2 : 두개의 sample 모두 Multithreaded applications으로 Oracle
v9.2.x에서만 존재합니다.
cpdemo1와 cpdemo2는 같은 내용을 실행하는데,
cpdemo2는 connection pooling을 이용하는 sample입니다.



Makefile의 구성 – Unix

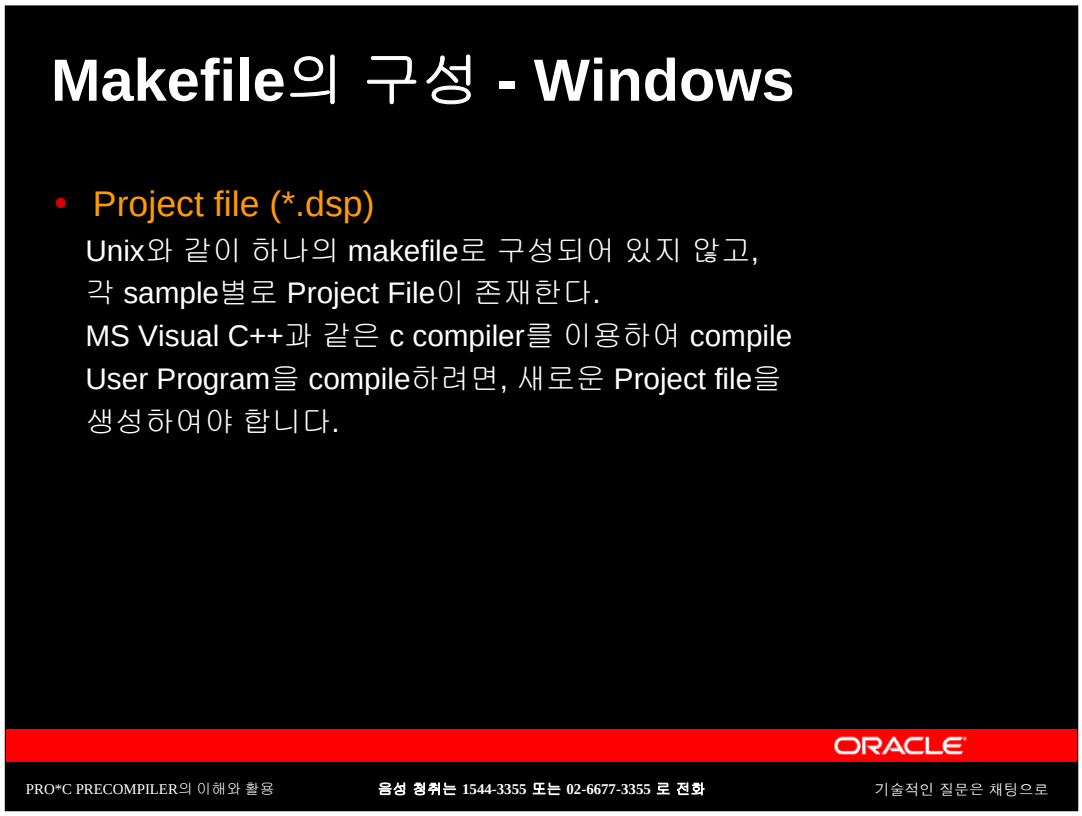
Pro*C를 precompile하기 위해서 Unix 장비에서는 make파일을 이용합니다.
이 make화일은 env_precomp.mk와 demo_proc.mk로 구성되어 있습니다.

• env_precomp.mk 파일

$ORACLE_HOME/precomp/lib에 위치하고, compile시에 필요한 모든 macro 변수들이 정의되어 있습니다.
각 DB와 OS Version별로 link하는 library와 c compiler option등이 정의되어 있습니다.
DB가 64bit인 경우, 64bit와 32bit별로 각각 나누어져 정의되어 있습니다.

• demo_proc.mk 파일

$ORACLE_HOME/precomp/demo/proc에 위치하고, DB가 64bit인 경우, demo_proc.mk, demo_proc32.mk, demo_proc64.mk로 각각 구성되어 있으며, env_precomp.mk를 include하여 필요한 macro 변수들을 가져다 사용합니다.
Pro*C sample을 compile할 수 있는 rule이 정의되어 있고, user program도 compile이 가능합니다.
다음과 같이 3가지 즉 precompile, c compile, link의 과정을 수행하여 최종적으로 실행 가능한 실행 화일을 만들어 주게 됩니다.



Makefile의 구성 - Windows

Windows의 경우, Unix와는 달리 make파일이 아니라, Project file이 설치됩니다.

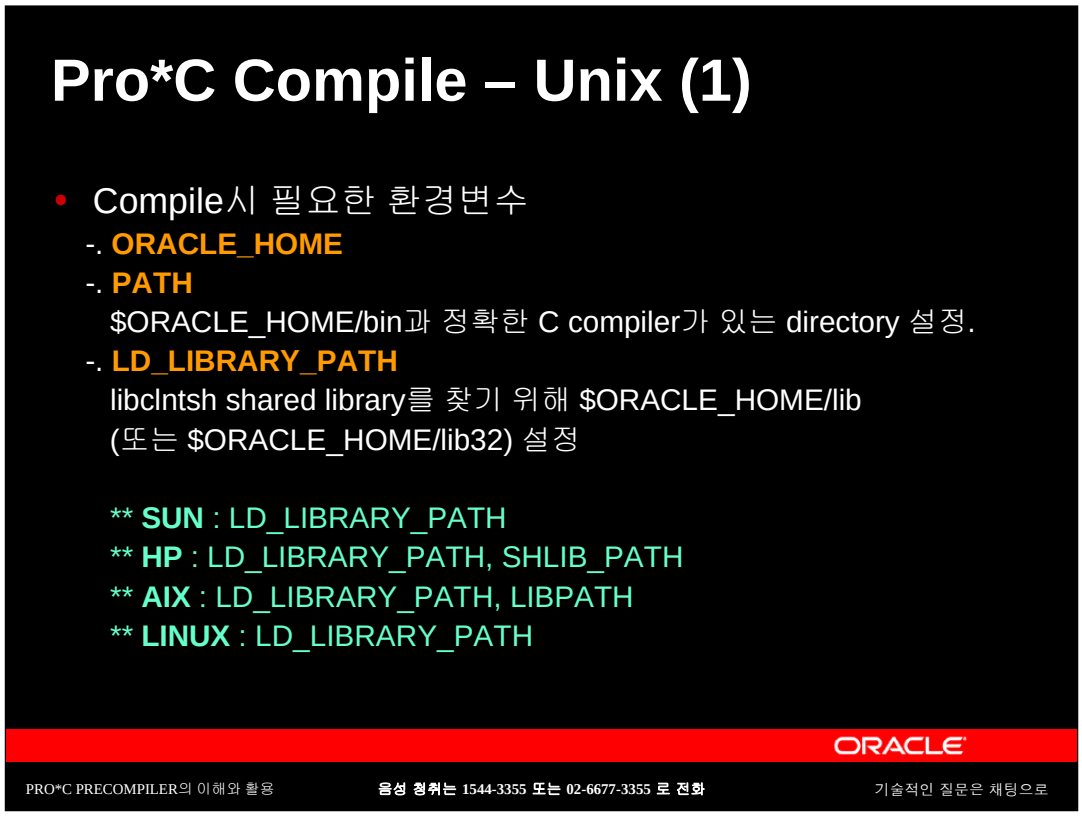
· Project file

Project file은 sample별로 각각 설치됩니다.

각각의 sample 디렉토리에 확장자가 *.dsp파일로 존재하며, 각 sample을 compile할 수 있습니다.

MS Visual C++과 같은 c compiler에서 이 project file을 load하여 compile하여야 합니다.

만일 새로운 User Program을 compile하려면, 새로운 project file을 생성하여야 합니다.



Unix 환경에서 Pro*C precompile

Unix에서 Pro*C를 Compile하기 위해 필요로 하는 환경 변수들이 있으며, 이런 환경변수를 Unix의 'env' command를 통해 정확히 확인 후 compile하여야 한다.

아래 환경변수 외에 ORACLE Database를 사용하기 위해 설정하는 NLS_LANG, TNS_ADMIN, ORA_NLS33 같은 변수들도 설정하여야 합니다.

- **ORACLE_HOME**

오라클 제품이 설치되어 있는 홈 디렉토리를 의미 합니다.

- **PATH**

PATH 환경변수에는 proc를 찾기 위해, $ORACLE_HOME/bin과 정확한 C Compiler가 있는 directory를 설정하여야 한다.

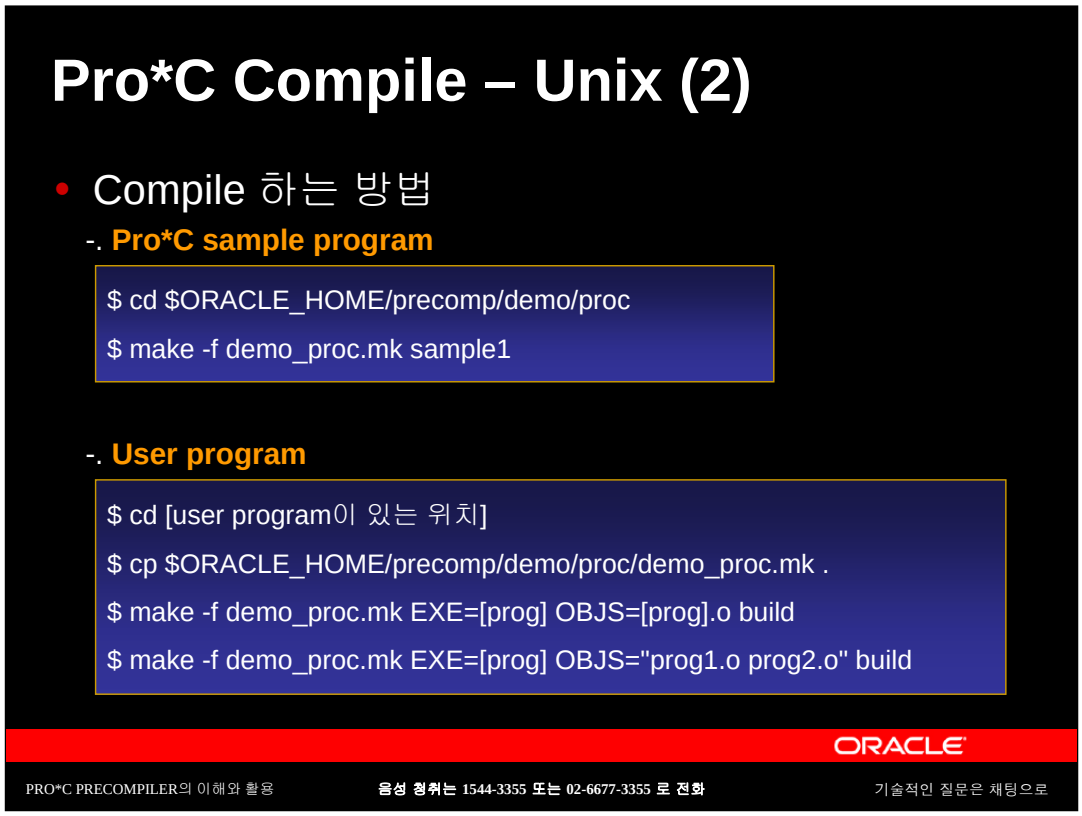
- **LD_LIBRARY_PATH**

libclntsh shared library를 찾기 위해 $ORACLE_HOME/lib를 설정하여야 한다. 설치되어 있는 오라클 제품이 64bit이고, 32bit library를 사용할 경우, $ORACLE_HOME/lib32로 설정하여야 합니다.

(Oracle 8i DB인 경우, $ORACLE_HOME/lib가 32bit이고, $ORACLE_HOME/lib64가 64bit이다.)

OS별로 다음과 같이 shared library를 찾는 환경변수가 조금씩 다르다.

- SUN, LINUX인 경우, LD_LIBRARY_PATH
- HP인 경우, LD_LIBRARY_PATH, SHLIB_PATH
- AIX인 경우, LD_LIBRARY_PATH, LIBPATH

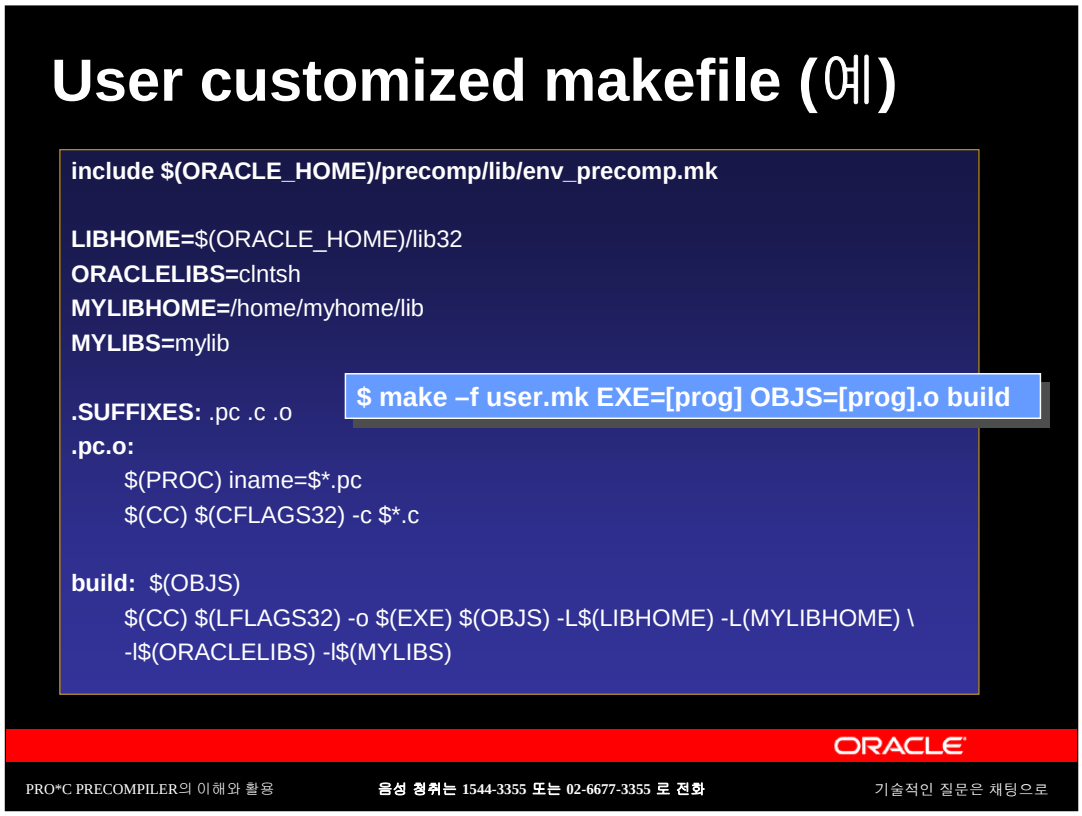


Unix환경에서 Pro*C를 precompile할때, demo_proc.mk을 이용하여 compile합니다.
Pro*C sample과 User가 생성한 Program에 따라서 아래와 같이 Compile하실 수 있습니다.

Pro*C sample program을 compile시에
$ cd $ORACLE_HOME/precomp/demo/proc
$ make –f demo_proc.mk sample1

User Program을 compile시에
$ cd [user program이 있는 위치]
demo_proc.mk를 현재 directory로 copy한다.
$ cp $ORACLE_HOME/precomp/demo/proc/demo_proc.mk .

object file이 하나인 경우,
$ make –f demo_proc.mk EXE=[prog] OBJS=[prog].o build
object file이 여러개인 경우,
$ make –f demo_proc.mk EXE=[prog] OBJS="prog1.o prog2.o" build

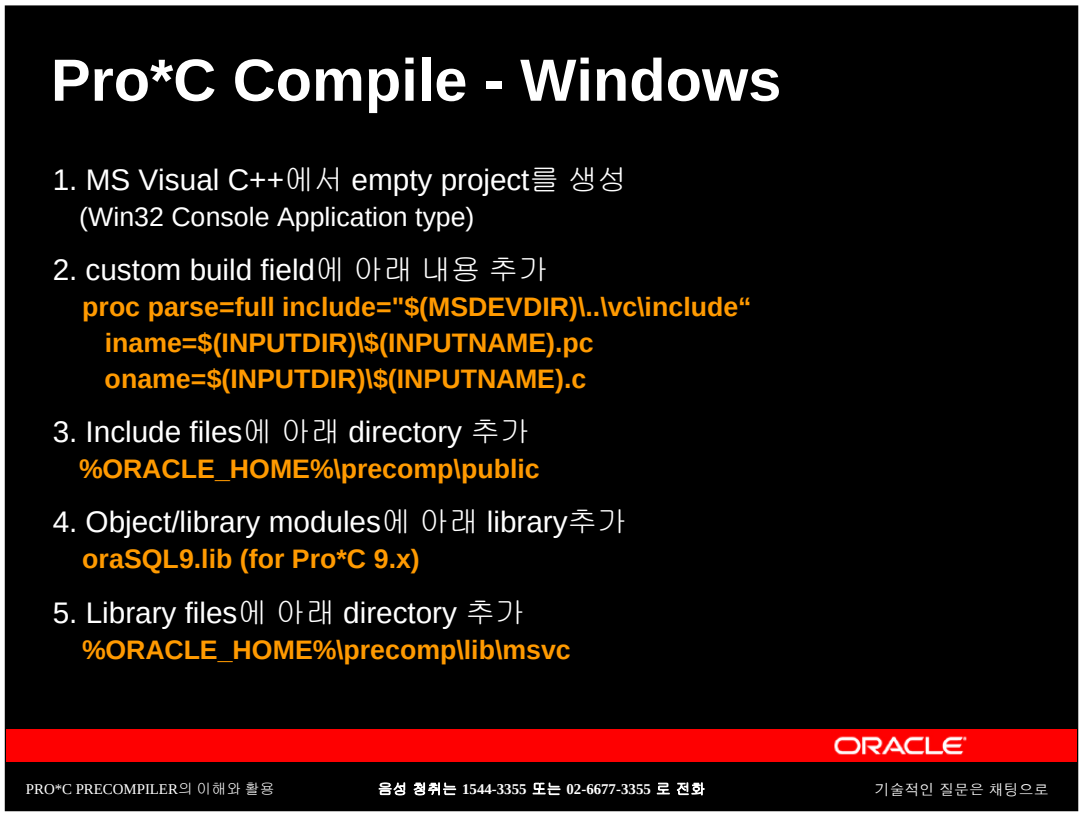


User makefile 예제

Pro*C를 compile하기 위해 demo_proc.mk를 이용하여 compile하실 수 있지만, User library와 다른 option들을 사용하기 위해서는 간단한 makefile을 생성하여 compile하는 것이 편리합니다.
위의 makefile은 env_precomp.mk를 include하여 32bit로 compile하는 예제입니다.

Pro*C는 Oracle에서 제공하는 shared library인 libclntsh가 항상 필요로 합니다.
물론, 다른 user library가 있다면, 위와 같이 MYLIBS로 선언 한 후, link를 하여 주시면 됩니다.
위의 CFLAGS32와 LFLAGS32 변수는 env_precomp.mk에서 가져와서 사용합니다.
위의 화일을 user.mk라고 하면, 아래와 compile하면 됩니다.

$ make –f user.mk EXE=[prog] OBJS=[prog].o build



Pro*C precompile – Windows

Windows환경에서 Pro*C를 compile하려면, MS Visual C++과 같은 compiler를 설치하여야 합니다.

Pro*C sample인 경우, 각 sample별로 존재하는 Project file을 MS Visual C++에서 load하여 Compile하면 됩니다.

User Program인 경우, 아래와 같이 Project file을 생성하여 Compile하여야 합니다.

MS Visual C++과 같은 compiler에서 새로운 project를 생성하고, 아래 내용들을 추가하여야 합니다.

'custom build' field에 precompile하는 command를 입력한다.

(proc parse=full include=...)

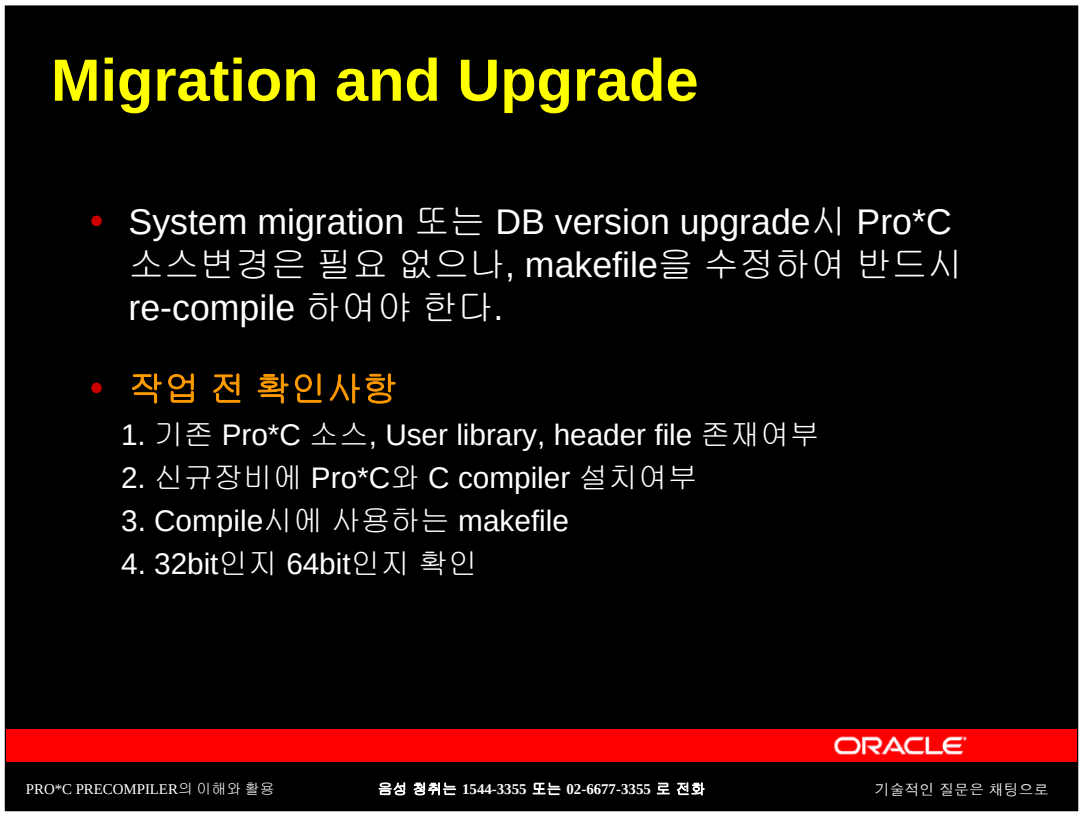
그리고, Tools -> Options -> Directories 메뉴에서 'Include files'란에 %ORACLE_HOME%₩precomp₩public를 넣어준다.

같은 방법으로, link tab에서 general category를 선택하고 'Object/library modules'에 oraSQL9.lib 파일을 추가한다.

그리고, 이 library file을 찾기 위해 Tools -> Options -> Directories 메뉴에서 'Library files'란에 %ORACLE_HOME%₩precomp₩lib₩msvc를 넣어준다.

위와 같이 Project file을 구성한 후, compile하면 됩니다.

좀 더 자세한 내용은 http://otn.oracle.co.kr 에서 기술문서 <bulletin :11970>를 참고하시면 됩니다.

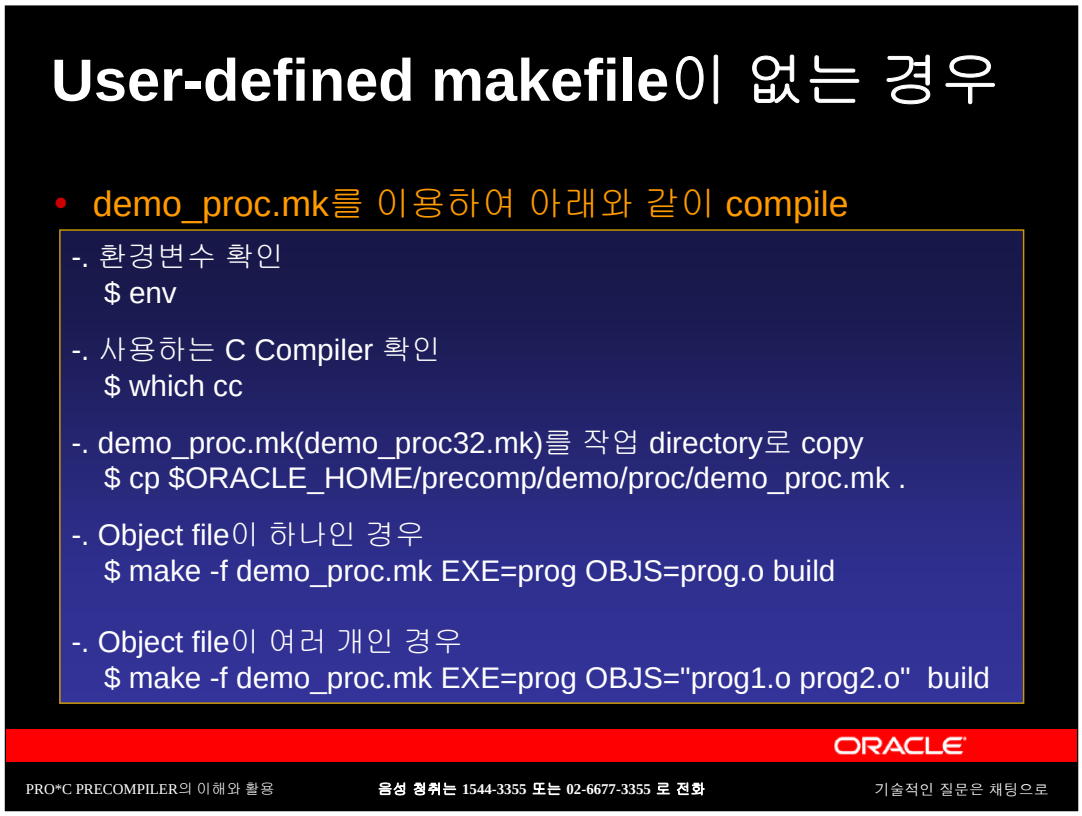


Migration and Upgrade

System migration이나, DB Version upgrade시에 Oracle library와 c compiler 가 변경됩니다. 그러므로, Pro*C를 re-compile하여야 합니다. 물론, Pro*C 소스변경은 필요 없습니다.

이런 작업에 앞서 아래와 같은 내용을 확인하여야 한다.

1. 기존 모든 Pro*C 소스와 user library, header file등이 존재하여야 한다.
2. 신규장비에 Pro*C와 적절한 C compiler가 설치되어 있어야 한다.
3. Compile시에 사용한 makefile이 있는지 확인한다. 만약, 존재하는 경우, 싶게 작업을 하실 수 있습니다.
4. 기존에 32bit인지, 64bit인지 확인하여야 하고, 신규장비에서 32bit로 compile할지, 64bit로 compile할지를 결정하여야 합니다.



User makefile이 없는 경우

기존에 사용하던 makefile이 없는 경우, Pro*C Program을 새로운 환경에서
다시 compile할 때, demo_proc.mk를 이용하여 아래와 같이 작업하시면 됩니다.

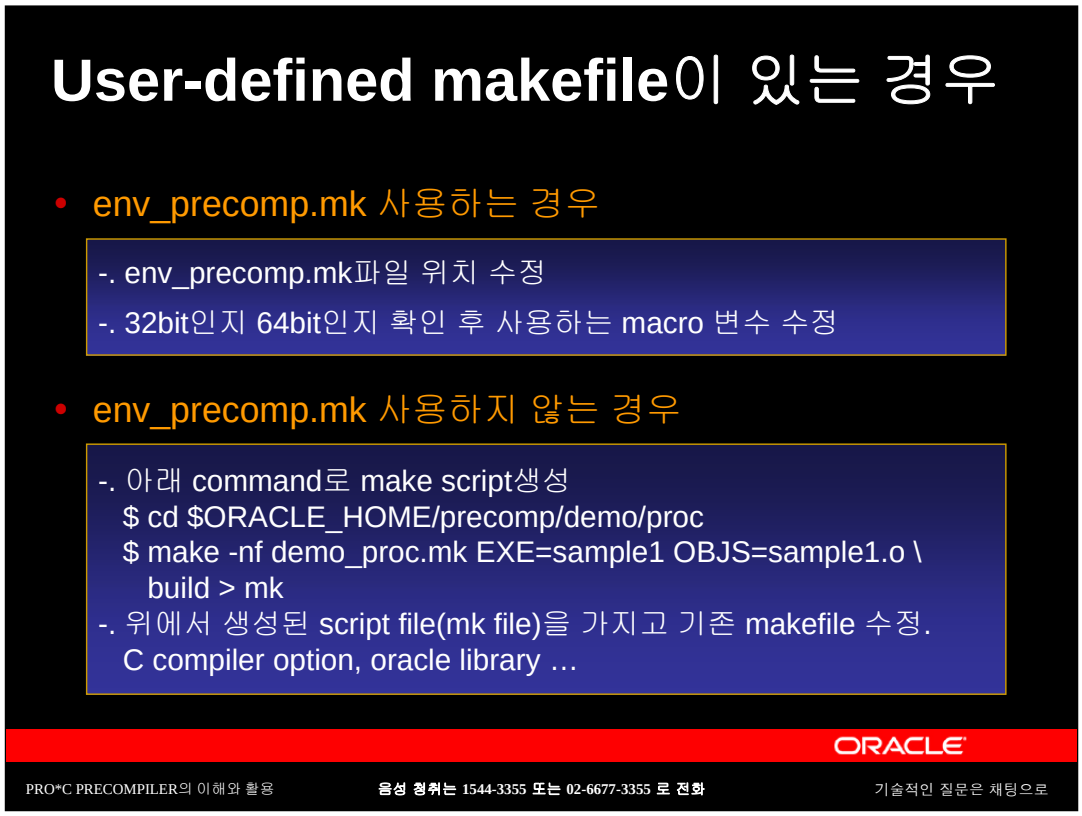
-. 환경변수를 확인한다.
'env' command를 통해, ORACLE_HOME, PATH, LD_LIBRARY_PATH등을 확인한다.

-. C Compiler를 확인한다.
'which cc'를 통해 PATH에 설정된 C Compiler를 확인한다.

-. Demo_proc.mk(또는 demo_proc32.mk)를 작업 directory로 copy한다.
'cp $ORACLE_HOME/precomp/demo/proc/demo_proc.mk .'

-. Object file이 하나인 경우,
'make –f demo_proc.mk EXE=[prog] OBJS=[prog].o build'

-. Object file이 여러 개 인 경우,
'make –f demo_proc.mk EXE=[prog] OBJS="[prog1].o [prog2].o" build'



User makefile이 있는 경우

다음으로, Pro*C program을 migration하거나 upgrade시에,
기존에 생성하여 이용하던 makefile이 있는 경우에 대해서 설명드리겠습니다.

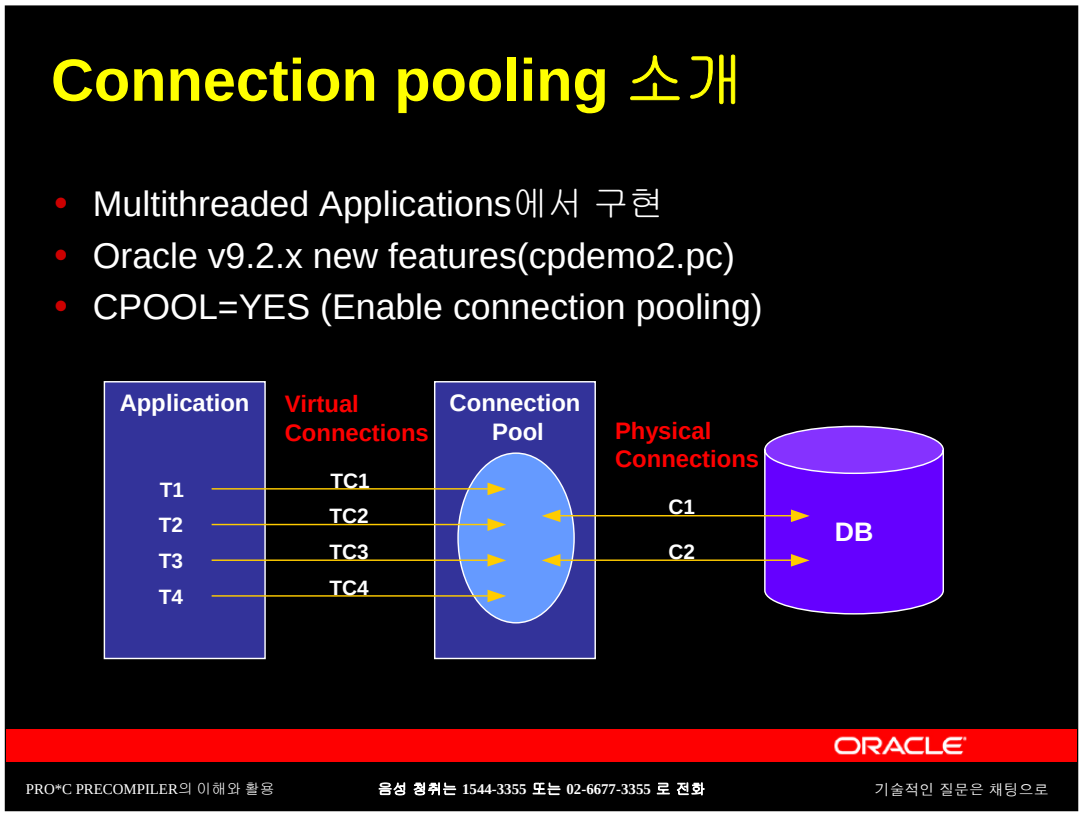
기존에 사용하는 makefile이 있는 경우, env_precomp.mk를 사용여부에 따라서 아래와 같이 작업을 하면 됩니다.

1. env_precomp.mk를 include하여 사용하는 경우,
   -. env_precomp.mk위치를 확인한 후, 파일 위치를 수정
   -. 32bit인지 64bit인지 Compile할지를 결정하여 사용하는 macro변수를 수정 후 compile.
   (CFLAGS32, LFLAGS32등을 수정하면 된다.)

2. env_precomp.mk를 include하여 사용하지 않는 경우,
   -. 아래 command로 compile하는 script를 확인한다.
   $ cd $ORACLE_HOME/precomp/demo/proc
   $ rm sample1 sample1.o sample1.c ➔ 파일이 존재하는 경우 삭제한다.
   $ make –nf demo_proc.mk EXE=sample1 OBJS=sample1.o build > mk

   -. 위에서 생성한 script file(mk file)에서 c compiler option, link하는 library등을 확인 한 후,
   기존 makefile을 수정하여 compile하면 됩니다.

** 결론적으로, User makefile 생성시 env_precomp.mk파일을 include하면,
System migration이나 DB server upgrade시에 User makefile을 간단하게 수정하실 수 있습니다.



Connection Pooling 소개

Pro*C Connection pooling을 사용하려면, DB Server가 9.2.x 이상 version을 사용하여야 하며, Multithreaded Application으로 작성하여야 합니다.

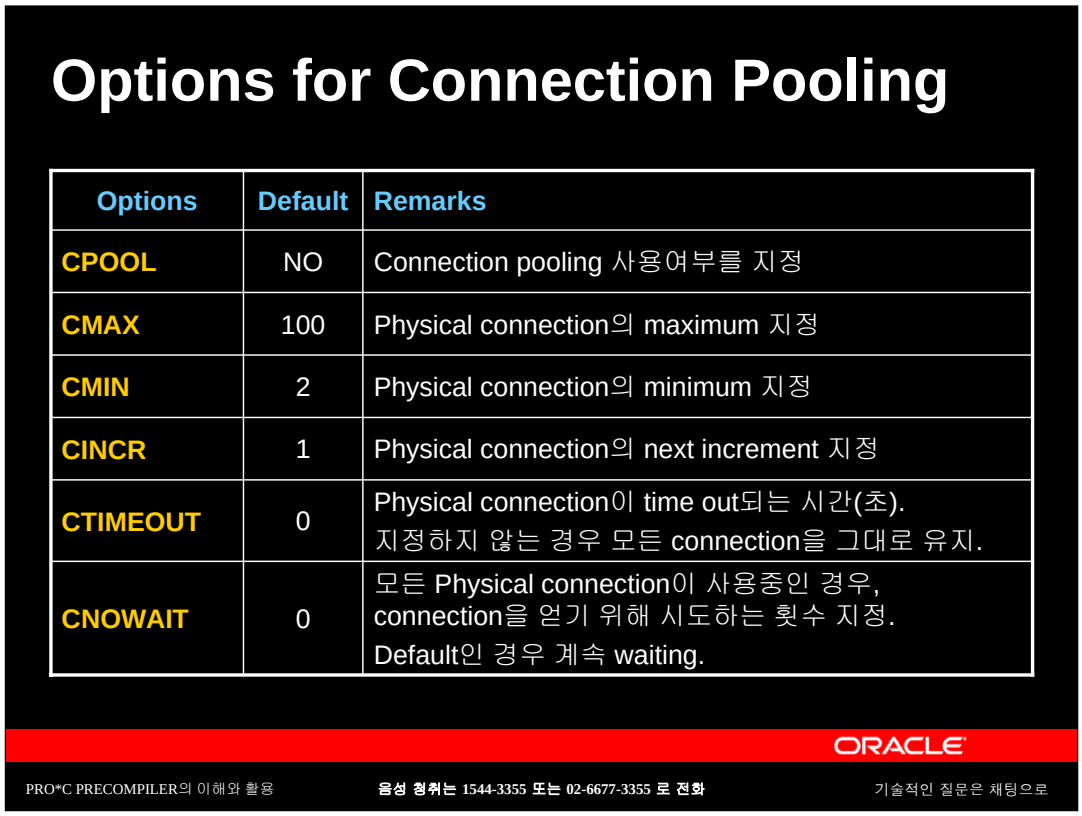
Connection pool을 사용하려면, 'CPOOL' option을 'YES'로 하고, Compile하면 됩니다.

위의 그림에서 하나의 Application이 Thread 'T1', 'T2', 'T3', 'T4'를 각각 생성하고, 각 Thread별로 DB와 연결하여 작업을 합니다.
이때, 'CPOOL' option을 'YES'로 설정하고 Compile하면, 이 Application은 Connection Pool을 이용하게 됩니다.

첫번째 Thread 'T1'에서 DB Connection 'TC1'를 요청할 때, SQLLIB는 Physical Connection인 'C1'으로 하나의 connection pool을 생성합니다.

두번째 Thread 'T2'에서 DB Connection 'TC2'를 요청할 때, Physical Connection 'C1'이 사용중이지 않으면, 'TC2'는 'C1'을 사용하게 됩니다. 반대로, 'C1'이 사용중이라면 새로운 Physical Connection 'C2'를 생성합니다.

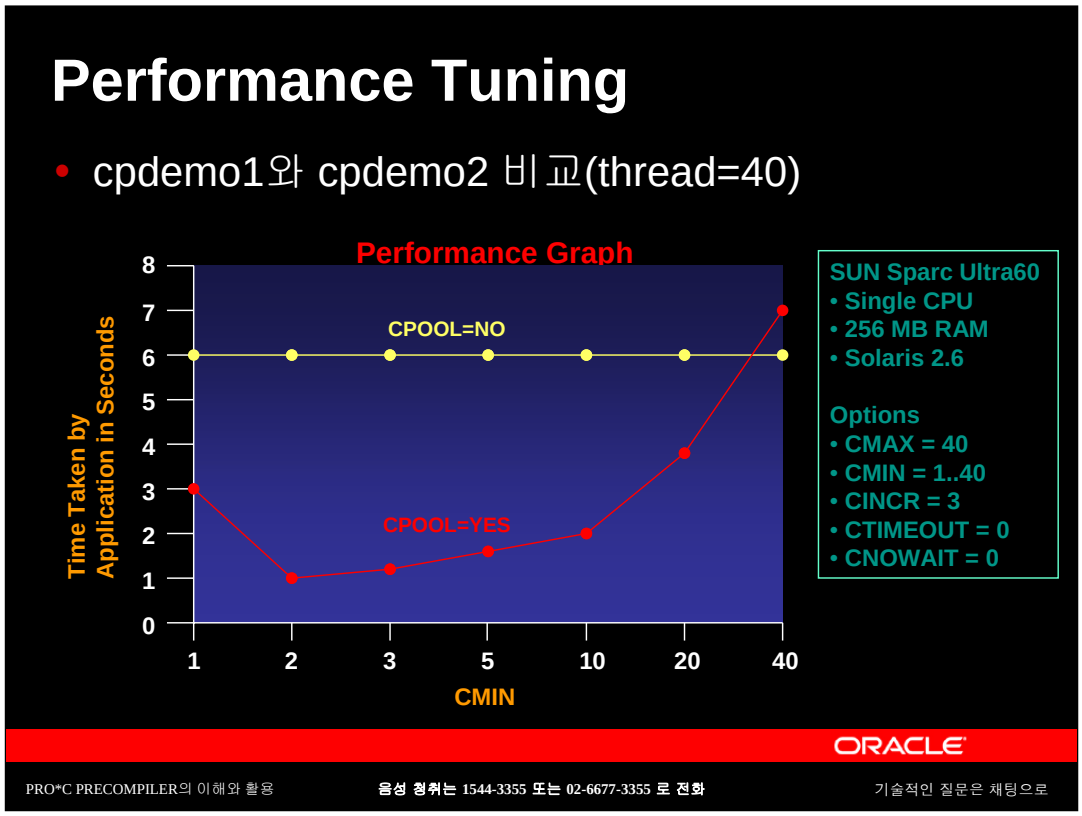
세번째 Thread 'T3' 에서 connection 'TC3'을 요청할 때, 'C1', 'C2'가 모두 사용 중이고, Max Connection이 '2'로 설정되어 있는 경우, Thread 'T3'은 waiting하던지, error가 발생하게 됩니다. Thread 'T4'도 Thread 'T3'과 같은 방법으로 처리됩니다.



Connection Pooling – Precompiler Option

Connection Pooling Option등을 이용하여 Min/Max Connection 수를 설정하고 idle timeout등을 설정할 수 있습니다.

• CPOOL option
connection pooling사용여부를 지정하고, default는 'NO'이다.
• CMAX option
Physical connection의 maximum을 지정한다. 'CMIN + CINCR' 보다 커야 한다.
• CMIN option
Physical connection의 minimum을 지정한다.
첫번째 Connection을 open을 할때, 이 Option에 지정된 수만큼 connection을 open한다.
• CINCR option
추가적인 Physical connection이 필요할 때, 증가크기를 지정한다.
현재 connection이 'CMAX'보다 적은 경우, 이 크기만큼의 connection을 추가적으로 open한다.
• CTIMEOUT option
Physical connection의 idle timeout을 초 단위로 지정한다.
지정하지 않는 경우, 모든 connection을 그대로 유지하게 된다.
• CNOWAIT option
모든 Physical connection이 사용중인 경우, connection을 얻기 위해 시도하는 횟수를 지정한다.
지정하지 않는 경우 계속 waiting한다. 반대인 경우, error message를 display한다.



Connection Pooling - Performance Tuning

Connection Pooling을 이용하여 Application의 처리속도를 향상시키는 방법에 대해 설명하겠습니다.

위 그래프는 Application이 처리되는 시간과 Min Connection에 대한 상관관계를 표시하였으며, Sample Program인 'cpdemo1'과 'cpdemo2'에서 Thread 40로 각각의 DB로 connection하여 작업 시에 처리되는 시간을 측정하였다.

Connection Pooling을 사용하지 않는 'cpdemo1'인 경우, Min Connection수에 상관없이 약 6초 정도 소요된다.

반면에, Connection Pooling을 사용하는 'cpdemo2'인 경우, Min Connection수에 따라서 약 1초에서 7초 정도 소요된다.

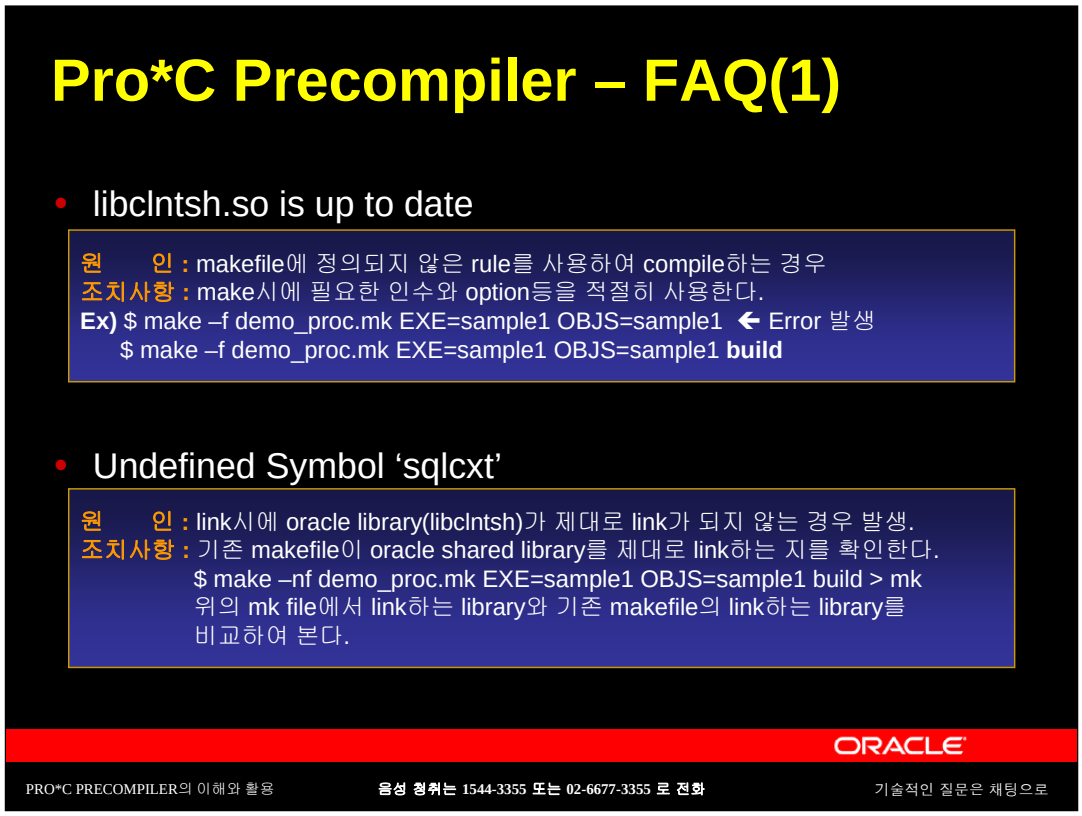
'cpdemo2'에서 Min Connection을 '2'로 설정하는 경우, 'cpdemo1'에 비해 처리속도가 1/6로 단축된다.

물론, Min Connection이 '40'인 경우, 'cpdemo1'보다 더 많은 시간이 소요됩니다.

그러므로, Connection Pooling을 사용하는 경우, Connection Pooling을 사용하지 않는 것 보다 훨씬 적은 시간이 소요됨을 알 수 있다.

하지만, Min Connection수에 따라서 반대의 현상이 발생할 수 있으므로, 충분한 Test를 통해 Connection Pooling option의 값을 적절히 설정하여야 한다.

*** 위의 처리속도는 Sun Sparc Ultra60, Single CPU, 256MB, Solaris 2.6 에서
CMAX=40, CMIN=1 .. 40, CINCR=3, CTIMEOUT=0, CNOWAIT=0 Option으로 Test



Pro*C Precompiler – FAQ

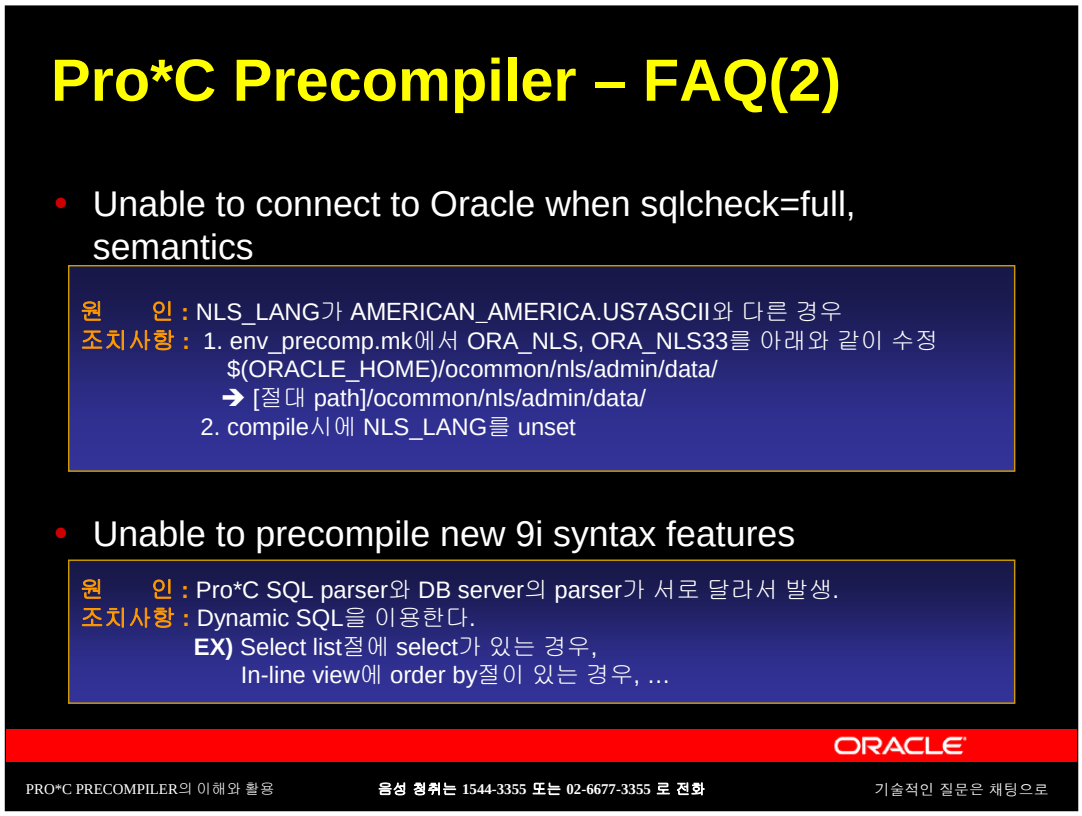
다음으로, Pro*C에서 자주 질문하시는 내용을 정리하였습니다.

• Libclntsh.so is up to date 에러

원 인 : makefile에 정의되지 않은 rule를 사용하여 compile하는 경우 발생한다.
조치사항 : makefile을 확인하여, Compile시 필요한 인수와 option등을 적절히 사용해야 한다.
예를 들면, $ make –f demo_proc.mk EXE=sample1 OBJS=sample1.o 라고 하면 error가 발생한다. 이 경우는 'build'를 사용하지 않아서 발생한다.

• Undefined Symbol 'sqlcxt' 에러

원 인 : link시에 oracle library(libclntsh)가 제대로 link하지 않는 경우 발생한다.
물론, symbol이 'sqlcxt'가 아닌 user가 정의한 symbol인 경우, user library가 link되지 않아서 발생한다.
조치사항 : 사용하는 makefile에서 oracle shared library를 제대로 link하는지를 확인한다.
$ make –nf demo_proc.mk EXE=sample1 OBJS=sample1 build > mk
에서 생성되는 Compile script를 통해 기존 makefile와 link하는 library를 비교하여 수정하면 된다.



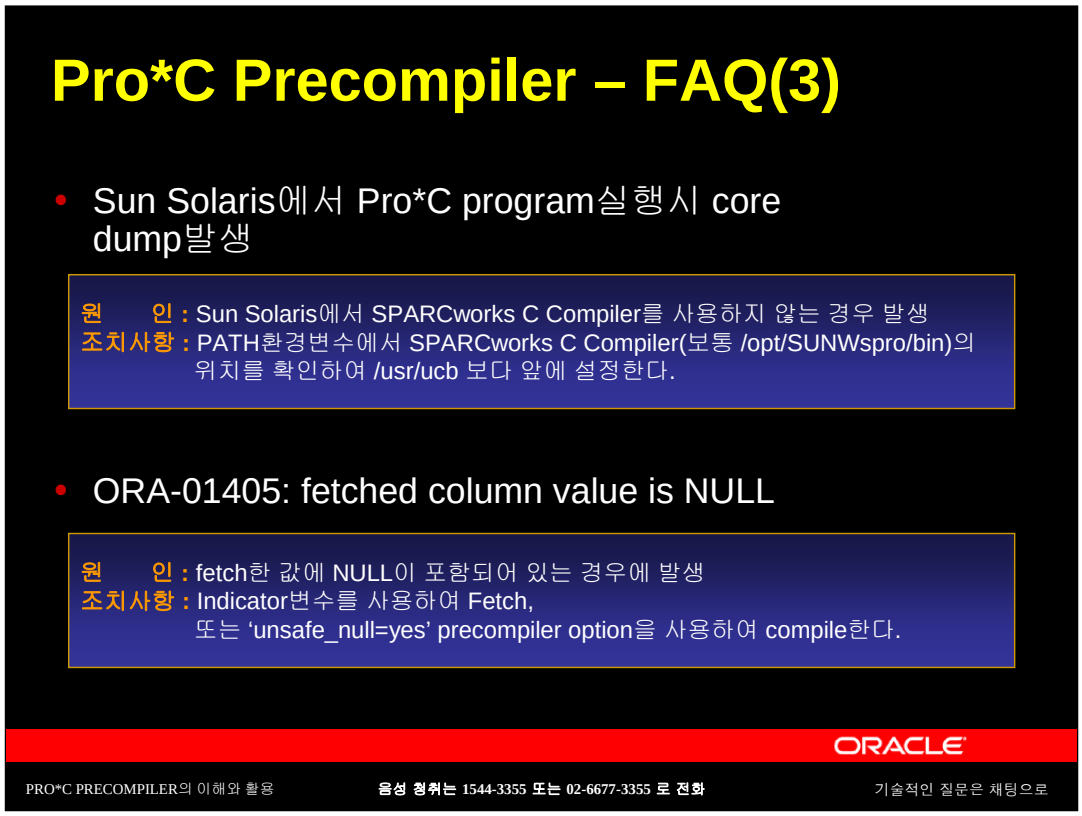
• sqlcheck=full, semantics를 사용하여 **Compile**할때, Unable to connect to Oracle 에러

원 인 : NLS_LANG이 American_America.US7ASCII와 다른 경우.
조치사항 :

1. env_precomp.mk에서 ORA_NLS, ORA_NLS33을 선언하는 부분이 있는데, 이 부분을 절대 PATH로 수정하면 된다.
2. 다른 방법은 compile시에만, NLS_LANG환경변수를 unset하여 준다.

• Oracle 9i에서 지원된 새로운 SQL syntax를 사용하는 경우, 에러 발생

원 인 : Pro*C의 SQL parser와 DB Server내의 SQL parser가 서로 달라서 발생합니다.
조치방법 : Dynamic SQL을 이용을 이용하여야 한다.

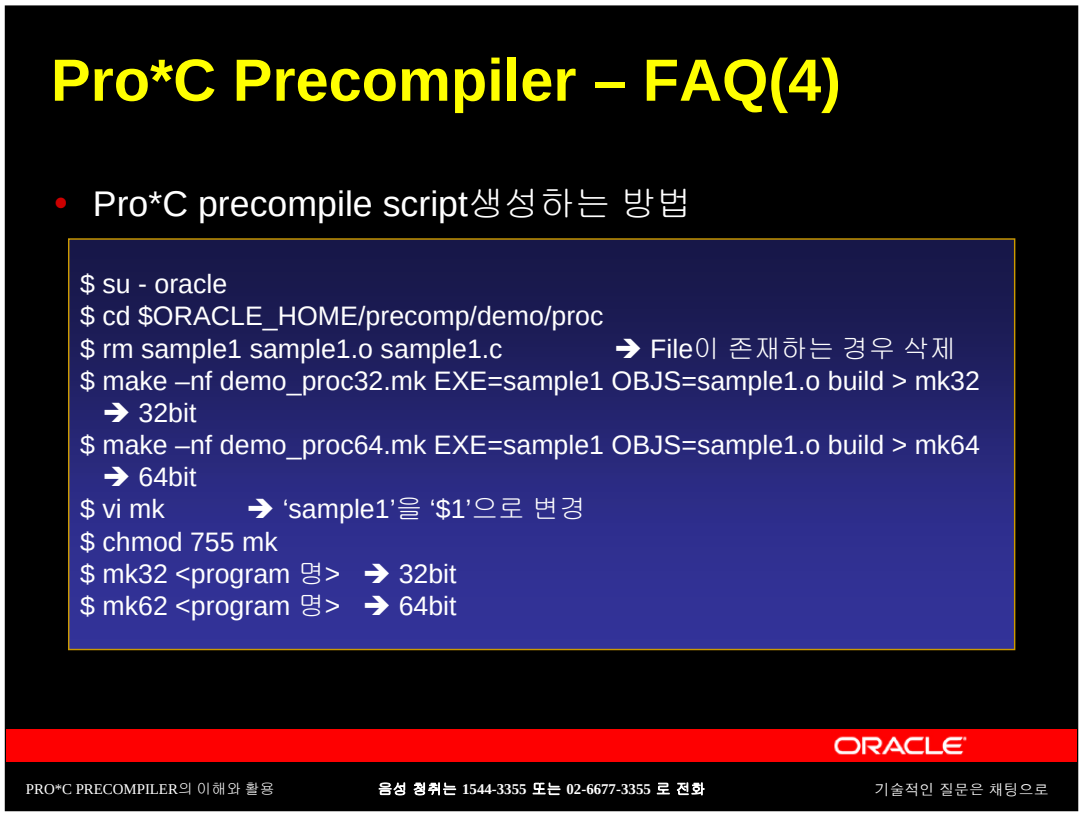


• Sun Solaris에서 Pro*C program실행시에 core dump발생

원 인 : Sun Solaris에서 /usr/ucb/cc 사용하여 compile한 경우 발생한다.
조치사항 : SPARCworks C compiler(보통 /opt/SUNWspro/bin)의 위치를 확인한 후, PATH환경변수에 /usr/ucb 보다 앞에 설정한다. 'which cc'를 이용하여 C compiler를 확인한다.

• ORA-1405 : fetched column value is NULL

원 인 : fetch한 값에 NULL이 포함되어 있는 경우에 발생한다.
조치사항 : Indicator변수를 사용하여 fetch한다. 또는 'unsafe_null=ues' precompiler option을 사용하여 compile한다.



• Pro*C compile하는 script를 생성하는 방법

$ su – oracle ➔ oracle user에서 실행한다.
$ cd $ORACLE_HOME/precomp/demo/proc
$ rm sample1 sample1.o sample1.c ➔ File이 존재하는 경우 삭제.

$ make –nf demo_proc32.mk EXE=sample1 OBJS=sample1.o build > mk32 ➔ 32bit
$ make –nf demo_proc64.mk EXE=sample1 OBJS=sample1.o build > mk64 ➔ 64bit

$ vi mk ➔ 'sample1'을 '$1'으로 변경

$ mk32 <program명> ➔ 32bit
$ mk64 <program명> ➔ 64bit