

To compile a C program using the Pro*C precompiler

1) **Get the demo_proc.mk from the Oracle proc/demo directory.**
/pkgs2/oracle8.0.4/app/oracle/product/8.0.4/precomp/demo/proc

2) **Write a Pro*C program and save it with the extension "pc".**

3) **Run the demo_proc.mk makefile to precompile and compile the program.**

make -f demo_proc.mk OBJS=fn.o EXE=fn build

(fn is the file name of your Pro*C program)

4) **Execute the program.**

There should be one like this for C++ programs using cppbuild instead of build.

```

/*
** This program shows a select imbedded in a C program
*/

#include <stdio.h>

EXEC SQL BEGIN DECLARE SECTION;
    VARCHAR uid[20];
    VARCHAR pwd[20];
    char  sid[9];
    char  sname[20];
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

int main()
{
    /* log into ORACLE */
    strcpy(uid.arr, "yourcli");
    uid.len=strlen(uid.arr);
    strcpy(pwd.arr, "yourpassword");
    pwd.len=strlen(pwd.arr);

    EXEC SQL WHENEVER SQLERROR GOTO errprt;
    EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
    if(sqlca.sqlcode == 0)
        printf("Connected to ORACLE user:%s\n", uid.arr);
    else
    {
        printf("Error occured. sqlcode = %d \n message = %s\n",
            sqlca.sqlcode, sqlca.sqlerrm.sqlerrmc);
        exit(0);
    }

    /* Open cursor associated with query */
    EXEC SQL DECLARE C1 CURSOR FOR
        SELECT STU_ID, STU_NAME
        FROM STUDENT;
    EXEC SQL OPEN C1;

```

```
/* Process answer set one at a time */  
EXEC SQL WHENEVER NOT FOUND GOTO finish;  
for(;;)  
{  
    EXEC SQL FETCH C1 INTO :sid, :sname;  
    printf("%-9s  %-20s\n", sid, sname);  
}  
  
/* When finished, close cursor, commit and get out */  
finish:  
  
EXEC SQL CLOSE C1;  
EXEC SQL WHENEVER SQLERROR CONTINUE;  
EXEC SQL COMMIT WORK RELEASE;  
  
exit(0);  
  
/* When an error occurs, rollback and get out */  
errprt:  
  
printf("\n %s \n", sqlca.sqlerrm.sqlerrmc);  
EXEC SQL ROLLBACK WORK RELEASE;  
exit(1);  
  
}
```

- **EXEC SQL declare <cursor-name> for <select-stmt>;**
Declares a cursor associated with an SQL query
- **EXEC SQL open <cursor-name>;**
- **EXEC SQL fetch <cursor-name> into <host-variable-list>;**
Retrieves the next row into the host variables
- **EXEC SQL close <cursor-name>;**
- **EXEC SQL begin declare section;**
Declaration of host variables. Host variables are used normally in C statements. In SQL statements they are preceded by a ‘:’.
- **EXEC SQL end declare section;**
- **EXEC SQL commit [work] [release];**
- **EXEC SQL rollback [work] [release];**

commit – makes database changes permanent.

rollback – reverses any changes made to the database.

release – disconnects from the database.

- **EXEC SQL whenever <condition> <action>;**

conditions

**sqlerror
sqlwarning
not found**

actions

**continue
goto <label>
stop
do function | break | return**

Embeds an ‘if’ after each ‘EXEC SQL’ statement to check for the condition and take the action in effect.

<condition> <action> stays in effect until overridden