# 한글 문서화일의 관리와 효과적인 코드변환에 관한 연구

윤호상 손진곤 황종선
고려대학교 전산과학과

# A Study on Hangul Text File Management and Effective Code Conversion

Ho Sang Yun Jin Gon Shon Chong Sun Hwang
Dept. of Computer Science, Korea University

## 요 약

한글 문서화일을 작성할 때 한글코드로서 일반적으로 많이 사용되는 것은 2 Byte 완성형 코드와 2 Byte 조합형 코드이다. 이 두 코드체계는 각각 내재된 장점이 있으며 이로 인하여 많은 논란이 되어 왔으나, 두 코드체계를 이용하여 작성되는 한글 문서 화일은 여전히 늘어나고 있는 것이 현 실정이다.

이러한 배경에서 한글 문서화일의 코드변환에 관해서는 계속 연구되어 왔고 현재 여러종류의 변환프로그램이 실용화 되어 있다.

본 논문에서는 기존의 변환프로그램에서 한글 문서화일을 화일 단위로 변환시키는 것을 표시난(tag field) 개념을 도입하여 디스크 단위로 변환시킬 수 있도록 개선된 변환프로그램을 제시하였다. 아울러 코드변환시 변환이 필요한 화일을 쉽게 찾을 수 있도록 한글 문서화일에 대하여 그 작성 코드의 종류를 표시할 수 있는 DOS명령어를 제시하였다.

## I. 서론

컴퓨터가 발명되어 개발되어온 환경이 주로 영어 사용 문화권인 이유로 컴퓨터가 처리할 수 있는 문자는 영어로 한정되어 있었다. 이러한 영어문화권의 컴퓨터를 우리나라에서 사용하기 위해서는 컴퓨터가 처리할 수 있는 한글 코드가 필요하게 된다. 그리고 한글 코드 체계를 개발하는데 있어서 오는 문제점은 우리 사회가 완전한 한글전용이 아니라 한글 이외에 한자, 영어를 동시에 사용하여야 한다는 것이다. 이와같은 문제점의 해결방안으로 여러 방법으로 컴퓨터에서 한글을 구현하기 위한 노력이 이루어졌으며 그 결과로 다양한 한글 코드가 생기게 되었는데 이것은 컴퓨터 기종사이의 호환성을 떨어뜨리고 작성된 한글문서에 대해서도 사용자가 보유하고 있는 컴퓨터의 기종에 따라 코드변환의 번거러움이 발생하였다[1,3].

이에 본 논문에서는 기존의 영문문서화일에 대하여 한글문서화일을독립적으로 관리함으로써 한글문서화일의 관리를 체계적으로 할 수 있고 컴퓨터 기종의 차이에 따르는 한글 코드의 변환을 효율적으로 해주는 방법에 대하여 연구하였다.

## II. 한글 코드

### 1. 종류 및 특징

지금까지 나온 한글 코드체계를 정리 하면 다음과 같다.

- n바이트 한글 코드
- 3바이트 한글 코드
- 7비트 완성형 코드
- 2바이트 조합형 코드
- 2바이트 완성형 코드

### 1.1 n바이트 한글 코드

n바이트 한글 코드는 멀티바이트 한글 코드라고도 하며 주로 터미널을 사용하는 대형 컴퓨터에서 사용하는 방법으로 각 자소 하나하나에 1바이트씩을 할당하는 방법이다.

이 한글 코드는 다시 7비트 n 바이트 한글 코드와 8비트 n 바이트 한글코드로 나누어지는데 7비트 한글코드체계는 한/영 구분을 SO/SI Delimeter를 사용하고 8비트 한글코드체계는 8번째비트(MSB)를 한/영 구분으로 사용한다[1,2,6].

1.2 3바이트 한글 코드

3바이트 한글 코드는 n 바이트 한글 코드의 단점을 보완한 방법으로 n바이트 한글 코드가 한글의 각 자소마다 1바이트를 할당한 것과 다르게 한글의 초성,중성,종성에 각각 1바이트씩을 할당한 것이다.

이 한글 코드 역시 7비트 3바이트 한글 코드와 8비트 3바이트 한글코드로 나누어지며 n바이트 한글 코드와 다른점은 종성이 없는 글자인 경우는 fill code라고 하는 가상문자를 채운다. 따라서 한글의 추가나 삭제시 간단하게 3바이트만 추가하거나 삭제하면 되므로 n바이트 한글 코드 체계보다는 개선된 방법이다[2,6].

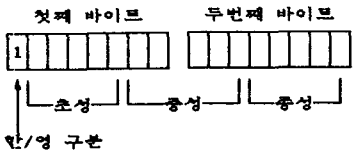
1.3 7비트 완성형 코드

이 한글 코드는 영어의 일반적인 표기 관례상 소문자 뒤에 대문자가 오지 않는다는 특성을 이용한 한글 코드로서 소문자 뒤에 이어 대문자가 오는 위치에 한글 자소를 넣어 한글을 구현하는 방법이다.

1.4 2바이트 조합형 한글 코드

이 한글 코드 체계는 n바이트 한글 코드체계가 가지는 한글을 처리할 때의 데이타의 낭비가 많은 단점(한글 한 음절을 위해 항상 3바이트를 할당하므로 종성이 없는 경우는 Fill Code 로 채우기 때문) 을 고려하여 16비트의 연속된 2바이트 코드를 5비트씩 묶어서 초성, 중성 ,종성으로 구분하여 사용하는 한글 코드체계이다[1,3]. 여기서 첫번째 바이트의 MSB(Most Significant Bit;최상위 비트)는 한글과 영문 을 구별하는데 사용되는 것으로 1로 세트되면 한글을 나타내고 0으로 세트되면 영문을 나타내게 된다.(그림 1참조)

<그림 1> 2바이트 조합형 한글 코드의 원리



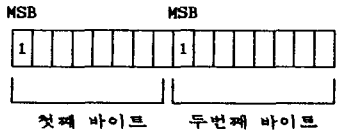

2바이트 조합형 한글 코드체계는 앞에서 언급한 한글 코드체계( n바이트 한글 코드, 3바이트 한글 코드)의 단점을 해결한 코드체계로서 7비트 코드체계에서 발생하는 문제점(한글/영문의 구별)을 해결했으며, n바이트 체계가 가지는 한글 음절에 대한 메모리의 양이 일정하지 않은 단점도 해결하였다. 그리고 이 한글 코드 체계는 한글의 구조적 특성을 가장 적합하게 살린 코드체계이다.

1.5 2바이트 완성형 한글 코드

2바이트 완성형 한글 코드는 앞에서 설명한 7비트 완성형 한글 보다 좀더 많은 문자를 수용할 수 있도록 개선된 한글 코드로서 1987년 3월 국가 표준안으로 지정된 규격으로 흔히 KS완성형 한글 코드 체계라고도 한다[4,6].

한글은 초성,중성,종성의 3성으로 구성되는데 이들 조합에 의해서 만들어진 글자 중에는 자주 이용되지 않는 글자의 수가 거의 70%에 이른다. 만들어 놓은 음절의 수가 2005자 정도면 누적 빈도율 99.999%가 되어 거의 모든 한글을 사용할 수 있다. 그래서 나온 방식이 완성형 한글 코드라고 할 수 있는데 이 방식은 통신이나 프로그램과 충돌하지 않는 부분에 한글을 음절 단위로 집어 넣었기 때문에 통신에 있어서는 아주 적합한 한글 코드라고 볼 수 있다.(그림 2참조)

<그림 2> 2바이트 완성형 한글 코드의 원리




2바이트 완성형 한글 코드는 현재 행정 전산망으로 이용되는 한글 코드가 이에 속하며 국가표준안에 의하면 한글 2350자, 한자 4880자와 기타 특수 문자 1128자를 선별하여 코드로 정의하고 사용자 정의 문자 또는 향후 예비 영역으로 470자를 비어두고 있다.

이러한 2바이트 완성형 한글 코드는 ISO (International Organization Standardization, 국제표준화 기구)의 부호 확장법을 따르고 있으며 제어코드 영역이나 유닉스 시스템에서 사용하는 메타 캐릭터(Meta Character)또는 와일드 캐릭터 같은 특수 문자들과 한글의 충돌을 피하도록 작성되었다[1,3,5].

2. 코드 변환의 필요성

87년 3월 한글 코드 표준안으로 2바이트 완성형 한글 코드가 채택 된 이후로 행정 전산망에서 이 한글 코드를 선택함에 따라 KS완성형 한글을 수용하는 한글 카드나 소프트웨어가 계속적으로 늘어가고 있다. 그러나 그 이전에 사용되어 오던 2바이트 조합형 한글 코드 체계를 따라 작성되는 한글문서 화일로 인하여 완성형 한글 코드를 선택한 기종과 기존의 조합형 한글 코드 체계를 따르는 기종사이의 정보 교환이 어렵게 되었다. 이러한 코드 체계의 상이성에 따른 호환성의 문제를 해결하기 위해서 한글 코드의 변환이 필요하게 되었다[3].

그리고 기존의 화일 시스템에서 한글이 사용된 문서와 영어를 사용한 문서를 같이 취급함으로써 한글 문서의 화일에 대한 체계적인 관리가 상당히 어려웠다. 그 이유로서는 한글 문서 자체가 국내 사정상 여러가지 한글 코드를 이용하여 작성된 관계로 작성된 한글 문서가 어떤 한글 코드로 작성된 것인지 알기가 어려웠기 때문이다. 따라서 작성된 한글 문서가 한글 문서인지 영문 문서인지를 판단하기 위하여 작성된 문서를 검색해야 하며 한글 문서인 경우에도 이 문서가 어느 한글 코드 체계를 따른 문서인지를 판단하기가 어려웠다.

이와 같은 문제점을 해결하기 위해서는 한글 문서를 영문 전용 문서와 구별할 수 있어야 하는데, 해결책으로서는 각각의 한글 문서에 추가적인 정보로서 이 문서

가 어떠한 한글 코드 체계를 따른 문서인가를 나타내 주는 정보를 유지하면 된다.

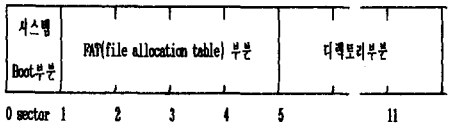
III. 한글 문서화일의 관리

1. 기존의 화일관리

기존의 DOS의 화일관리는 플렉시블 디스크상에서 화일의 관리를 뜻하는데 여기서는 요즘 PC에서 많이 사용되고 있는 MS-DOS에서의 화일관리를 어떻게 하는지를 알아보자.

MS-DOS에서의 화일 관리는 화일의 이름과 그 화일이 저장되어 있는 위치를 별개의 테이블에서 관리하고 있다. 즉, 일반적으로 디렉토리라고 하는 곳에는 화일 이름과 화일의 저장 위치를 표시하는 테이블의 엔트리 및 그 화일 각각의 속성이 저장되어 있고 이 디렉토리에 대하여 화일이 저장되어 있는 장소를 관리하는 FAT(file allocation table)라고하는 테이블을 유지한다(그림 3참조).

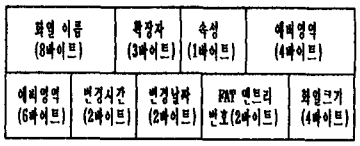
<그림 3> 플렉시블 디스크의 시스템 영역




1.1 디렉토리 엔트리(directory entry)

디렉토리 안에는 화일의 이름이나 하위 디렉토리 이름 이외에도 각각에 대하여 여러가지 정보를 유지하고 있는데 이것이 "디렉토리 엔트리"이다[7](그림 4참조).

<그림 4> 디렉토리 엔트리




디렉토리 엔트리는 총 32바이트를 영역을 사용하는데 이 영역은 위의 그림 4와 같이 8개의 부분영역으로 나누어진다.

- 화일 이름(file name)

이 부분은 화일의 이름이나 하위 디렉토리의 이름을 저장하는 부분으로 8개의 바이트를 사용한다.

- 화일의 속성(file attribute)

이 부분은 1바이트를 차지하는데 이 바이트가 화일의 속성을 나타내 준다.

- 변경시간과 변경 날짜

이부분은 각각 2바이트를 차지하여 화일이 갱신되거나 생성된 날짜와 시간을 저장하는 부분이다.

- FAT 번호

이 부분은 2바이트를 차지하며 화일이 저장되어 있는 위치를 나타내 주는 FAT의 시작위치를 나타내 준다.

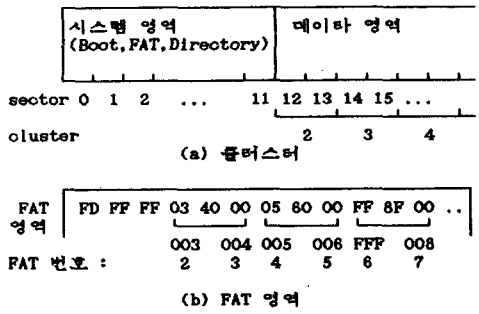
- 화일의 크기

이부분은 4바이트를 차지하며 화일의 크기를 나타내 준다.

1.2 FAT(file allocation table)

이 부분은 화일의 위치를 나타내주는 테이블로서 여기서는 디스크에서 위치를 나타낼 때 트랙(track)이나 섹터(sector)를 사용하지 않고 클러스터(cluster)라는 개념을 사용하는 데 이 클러스터는 몇개의 섹터를 하나로 묶어놓은 논리적단위이다. 보통 2개의 섹터가 1개의 클러스터에 대응된다[7](그림 5참조).

<그림 5> 클러스터와 FAT 영역




2. 한글 문서화일의 관리

화일의 관리에 있어서 OS는 화일의 이름(file name)과 확장자(extentin), 그리고 기타 정보(생성일,시, 화일의 크기, 속성 등) 을 유지하는데 이와 같은 정보에 한글 화일에 대한 정보(한글문서와 영문문서의 구분,작성된 한글코드)가 추가 되어야 하는데 이와같은 한글 문서화일에 대한 정보의 유지는 기종 간에 서로 다른 한글 코드를 사용하므로써 나타나는 한글 사용의 호환성 문제를 해결하기 위한 한글 코드의 변환에 아주 중요한 정보를 제공한다.

이와 같은 한글 문서 화일에 대한 정보를 유지하는 방법으로는 다음과 같은 방법들이 있다.

- 한글 문서화일을 위한 목록 화일 유지
- 화일의 속성을 나타내는 비트중에서 사용되지 않는 비트(표시난(tag field)의 사용

2.1 한글 문서화일의 관리를 위한 목록화일의 유지

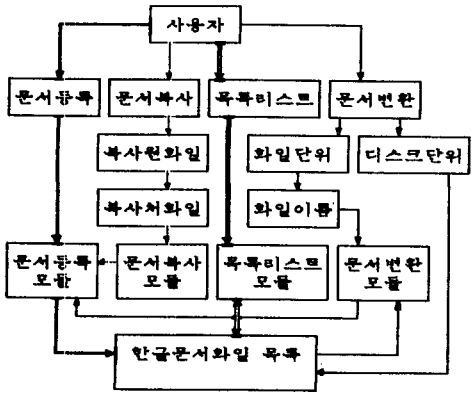
한글 문서화일의 관리를 위한 목록화일을 유지하는 방법은 기존의 OS에 의존하지않고 한글 문서화일을 관리할 수 있는 방법으로 한글 문서화일을 관리하는 통합적인 소프트웨어 패키지를 사용하는 방법이다.

이 방법은 한글문서의 목록을 유지하는 화일을 관리하는 화일 관리 프로그램이라고 볼 수 있는데 이 프로그램은 다음과 같은 기능을 가져야한다.

- 한글문서화일의 등록
- 한글문서화일의 정보 유지
- 한글문서화일의 복사
- 한글문서화일의 목록 표시
- 한글문서화일의 코드변환

이와같은 기능을 가지는 프로그램으로 한글문서화일의 관리에 종합적인 관리환경을 사용자에게 제공할 수 있다(그림 6참조). 그러나 이 방법을 사용할 경우 발생하는 문제점으로서는 결론적으로 한글문서화일의 관리와 영문화일의 관리가 완전히 분리된다는 것이다. 즉, 한글문서화일에 대해서는 반드시 이 프로그램을 통해서 관리해야 하며 영문화일은 기존의 화일관리 명령을 이용해야한다. 화일 관리의 2분화는 사용자에게 컴퓨터의 사용에 있어서 복잡성을 증가시키게 된다.

<그림 6> 한글문서화일의 관리를 위한 목록화일의 유지

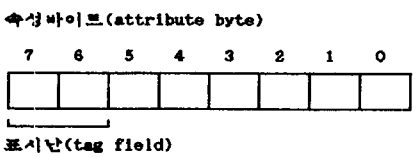



2.2 표시난(tag field)의 사용

이 방법은 프로그램 수준의 한글 문서화일 관리 방법의 단점을 보완한 방법으로서 기존의 OS의 명령어들을 그대로 사용하면서 약간의 명령어만 추가시켜 한글문서화일의 관리를 할 수 있도록한 방법이다.

기존의 OS는 화일의 이름과 속성 등의 정보를 유지하는 부분 즉, 디렉토리 엔트리를 가지고 있는데 이 중에서 화일의 속성을 나타내는 속성바이트(attribute byte)가 있다. 속성바이트에서 0 - 5비트는 화일의 속성을 나타내기 위해서 사용되고 6,7비트는 사용되지 않고 있다. 우리는 속성바이트 중 사용되지 않고 있는 6,7비트 부분을 표시난(tag field)이라 부르며 이것을 이용하여 한글 문서와 영문 문서를 구분할 뿐만 아니라 한글 문서화일의 작성에 사용된 한글 코드의 종류를 기록할 수 있다(그림 7참조).

<그림 7> 표시난(tag field)의 할당 위치




이 방법은 기존의 한글 문서화일에 대하여 한글 정보를 셋팅하는 명령어와 한글 문서화일을 관리할 수 있도록 디스크나 디렉토리내의 한글 문서화일의 목록을 나타내주는 명령어가 필요하게 된다.

표시난을 둠으로써 한글 문서화일을 효율적으로 관리할 수 있을 뿐 아니라 한글 문서화일의 코드변환에 있어서도 기존의 화일 단위이외에도 디스크나 디렉토리단위로 한글 문서를 변환할 수 있다. 이에 관한 예는 IV장에서 설명하겠다.

## IV. 표시난(tag field)의 활용 예

이 장에서는 표시난(tag field)을 이용하여 한글 문서화일을 관리할 때 필요한 명령어(HSAVE,HDIR)와 한글변환프로그램(HCONVERT)을 설명하겠다.

### 1. 한글 코드 셋팅 --- HSAVE

한글 문서화일에 대해 그 화일을 작성할 때 사용된 한글 코드의 종류를 표시난에 셋팅시키는 명령어로서 명령어의 형식은 다음과 같다.

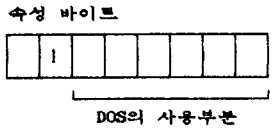
```
명령형식 :   HSAVE file-name/code-name
```

. file-name : 한글 문서 화일 이름
. code-name : 한글 코드가 완성형인 경우 f
한글 코드가 조합형인 경우 c

이 명령어는 한글 문서에 대한 정보를 유지하는 부분(디렉토리 영역의 0A번지) 에 기존의 화일이나 새로운 화일에 대한 정보를 세팅시킨다. code-name이 c인 경우 즉, 한글문서가 조합형인 경우는 6번째 비트를 셋팅하고(그림 8.a), code-name이 f인 경우 즉, 한글문서가 완성형인 경우는 7번째 비트를 셋팅한다.(그림 8.b)

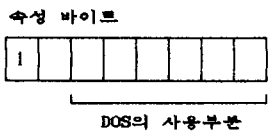
<그림 8> code-name에 따른 표시난의 셋팅

code-name : c




(a) 조합형인 경우

code-name : f




(b) 완성형인 경우

다음 그림 9는 HSAVE명령을 사용한 예이다.

<그림 9> HSAVE 명령의 예

```
A>hsave korea.dat/c  :  한글문서화일  korea.dat를
ATTRIBUTE SETTED !!!    조합형 한글문서로 셋팅
A>hsave seoul.dat/f  :  한글문서화일  seoul.dat를
ATTRIBUTE SETTED !!!    완성형 한글문서로 셋팅
A>hsave test.dat/f
ATTRIBUTE SETTED !!!
```

(a) 모니터에 나타나는 HSAVE명령의 결과

```
================ Hex codes ================ ASCII value
48 44 49 52 20 20 20 20 43 4F 4D 20 00 00 00 00 00 00 HDIR    COM ──── 일반 영문 화일
00 00 00 00 00 00 87 A2 14 13 02 00 E0 0E 00 00
4B 4F 52 45 41 20 20 20 44 41 54 A0 00 00 00 00 00 00 KOREA   DAT ──── 조합형 한글화일
00 00 00 00 00 00 19 5B 16 13 06 00 FF 18 00 00
48 53 41 56 45 20 20 20 43 4F 4D 20 00 00 00 00 00 00 HSAVE   COM
00 00 00 00 00 00 6F 64 16 13 00 00 AB 0F 00 00
53 45 4F 55 4C 20 20 20 44 41 54 60 00 00 00 00 00 00 SEOUL   DAT ──── 완성형 한글화일
00 00 00 00 00 00 19 5B 16 13 11 00 FF 18 00 00
54 45 53 54 20 20 20 20 44 41 54 60 00 00 00 00 00 00 TEST    DAT
00 00 00 00 00 00 19 5B 16 13 74 00 FF 18 00 00
```

(b) 그림 (a)에 대한 디렉토리 영역의 덤프 리스트

2. 한글 문서화일의 목록 리스트 명령 --- HDIR

이 명령은 등록된 한글문서화일의 리스트를 화면에 표시해주는 명령으로 앞에서 설명한 HSAVE명령을 이용하여 한글문서화일을 등록한 후 등록된 문서화일을 관리할 때 사용하는데 그 형식은 다음과 같다.

| 명령형식 : HDIR [D:d\] |
|---|

. D : 드라이브 이름
. d : 디렉토리 이름

이 명령어는 DOS의 DIR명령과 유사한 명령어로 지정된 드라이브의 디렉토리에 있는 한글 문서에 대한 정보를 화면에 표시해준다.

즉, HDIR명령은 시스템의 디렉토리 부분의 디렉토리 엔트리를 읽어서 한글 문서화일로 셋팅되어 있는 화일만을 골라 화일 이름과 그 화일의 작성에 사용된 한글 코드를 나타내 준다.

다음 그림 10은 기존의 DIR명령과 HDIR명령을 나타낸 것이다.

<그림 10> DIR명령과 HDIR명령

```
A>dir

 Volume in drive A has no label
 Directory of  A:\

HDIR     COM     3808   8-20-89   8:20p
KOREA    DAT     6399   8-22-89  11:48a
HSAVE    COM     4011   8-22-89  12:35p
SEOUL    DAT     6399   8-22-89  11:48a
KOREA    CVT     2048   1-01-80  12:06a
S        ASM     4246   8-22-89  12:35p
```

(a) DIR 명령

```
A:\>hdir
===  Hangul Management Directory ===
filename ext        code
------------------------
KOREA    DAT        C
SEOUL    DAT        F
TEST     DAT        F
```

(b) HDIR 명령

3. 한글 문서 코드 변환 프로그램 --- HCONVERT

한글 문서에 대한 정보를 유지함에 따라 한글 문서에 대한 원하는 코드 형태로의 코드 변환을 간편하게 할 수 있다. 기존의 코드 변환 프로그램들은 화일 단위로 한글 코드를 변환시켰기 때문에 우리는 원하는 한글 문서를 검색하여 이 문서가 어떠한 한글 코드로 작성되었는지를 알아낸 후 한글 변환 프로그램을 적용시켜야 한다.

그러나 한글 문서에 대한 정보를 유지함으로써 한글 문서에 대한 한글 코드변환을 디스크 단위나 디렉토리 단위로 할 수 있으므로 시간과 노력을 절약할 수 있다. 개선된 한글 코드 변환프로그램의 사용 형식은 다음과 같다.

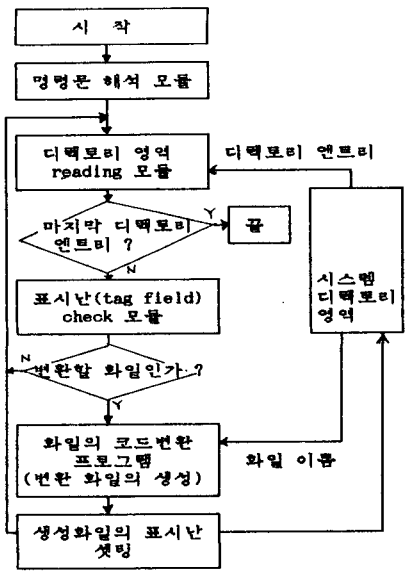
| 형식 : HCONVERT/code-name [D:d\] |
|---|

. D:d : 한글 코드변환을 원하는 디스크이름과 디렉토리 이름
. code-name : 바꾸고자하는 한글 코드

이 프로그램은 디스크 D의 디렉토리 d 내의 한글 한글 문서 중에서 code-name과 다른 한글 코드를 사용한 문서들만 code-name의 한글 코드로 변환시켜 확장자 CVT를 가지는 새로운 화일을 만든다.

그림 11은 프로그램의 실행과정을 나타낸다.

<그림 11> 한글 코드 변환 프로그램 실행과정




<그림 12> HCONVERT 프로그램의 실행 예

```
A:\>hdir
===  Hangul Management Directory ===
filename ext        code
------------------------
KOREA    DAT        C
SEOUL    DAT        F
TEST     DAT        F
```

(a) 실행전 한글문서 리스트

```
A:\>hconvert/c
CONVERSION COMPLETE ...
CONVERSION COMPLETE ...
```

(b) 실행

```
A:\>hdir
===  Hangul Management Directory ===
filename ext        code
KOREA    DAT        C
SEOUL    DAT        F
TEST     DAT        F
SEOUL    CVT        C
TEST     CVT        C
```

(c) 실행 후 한글문서 리스트

## V. 결론

본 논문에서는 현재 국내에서 사용되고 있는 여러 한글 코드에 대하여 조사했으며 이와 같은 다양한 한글 코드의 사용에 따르는 문제점을 조사한 결과 한글 문서화일의 체계적인 관리와 한글 코드변환이 필요하였다.

한글 문서화일의 체계적인 관리와 효과적인 코드변환을 위해서 프로그램 수준 및 OS수준에서 연구,검토 하였으며 특히 OS수준에서의 한글 문서화일의 관리에 대하여 중점적으로 연구 하였다. 이에 따라 추가될 새로운 명령어 HSAVE, HDIR과 새로운 한글 코드변환 프로그램 HCONVERT를 연구,구현하였다.

## VI. 참고문헌

1. 과학 기술처,한글 정보 처리 표준화 연구, 1986.7
2. 과학 기술처,한글 정보 처리 표준화 연구(3차년도) 1988.7
3. 과학 기술처, OS한글화 지침서 개발에 관한 연구, 1987.7
4. 한국전자 통신 연구소,한글 코드 표준화, 1987.12
5. 오길록, 박세영 "한글 정보처리를 위한 기초 연구" 정보과학회지 2권 4호 pp 7-19, 1984.12
6. 황대길 " 한글 자료 처리용 부호계 " 정보과학회지 2권 4호 pp 38-41, 1984.12
7. Microsoft Corp., MS-DOS User manual Ver 3.30, 1989.