

# Disclaimer

- The material provided in this document is not my original work and is a summary of some one else's work(s).
- A simple Google search of the title of the document will direct you to the original source of the material.
- I do not guarantee the accuracy, completeness, timeliness, validity, non-omission, merchantability or fitness of the contents of this document for any particular purpose.
- Downloaded from [najeebkhan.github.io](https://najeebkhan.github.io)



## Chapter # 4: The Operating Environment



**HMM TOOL KIT HTK**




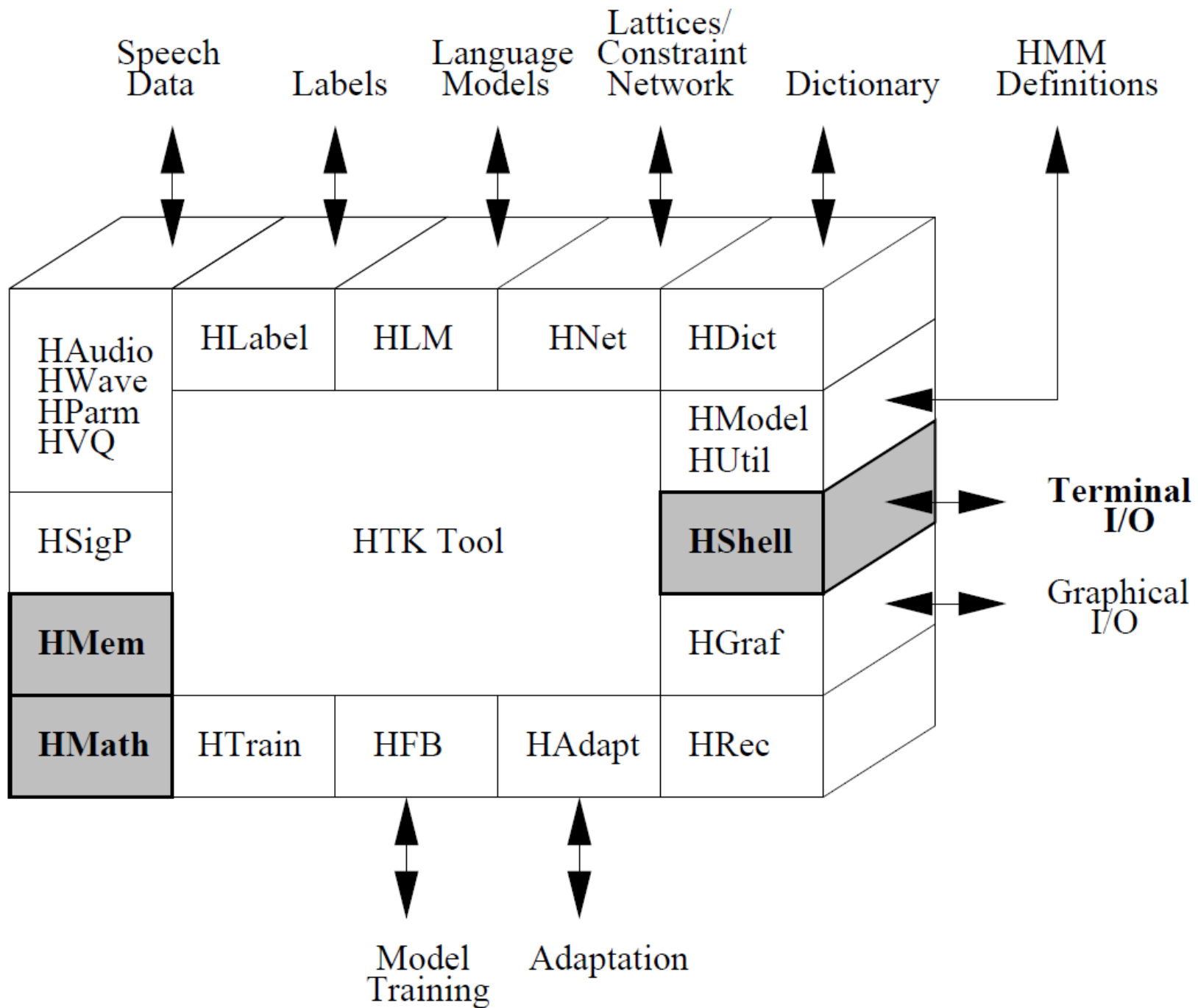
# Outline

- ✓ Introduction
- ✓ The Command Line
- ✓ Script Files
- ✓ Configuration Files
- ✓ Standard Options
- ✓ Error Reporting
- ✓ Strings and Names
- ✓ Memory Management
- ✓ Input / Output via Pipes and Networks
- ✓ Byte-swapping of HTK data files



# Introduction

- The behavior of a HTK tool depends on three sources of information
    - All HTK tools are executed by issuing commands to the operating system shell
    - Setting parameters in a configuration file
    - A small number of parameters are specified using environment variables
- 



# The Command Line

- The general form of command line for invoking a tool is

```
tool [options] files ...
```

-i	- a switch option
-t 3	- an integer valued option
-a 0.01	- a float valued option
-s hello	- a string valued option

```
HList s1 dir/s2 /users/sjy/speech/s3
```

```
HInit -L mymodels/new/ hmmfile data*
```

```
HInit -L mymodels/new hmmfile data*
```

# Script Files

- Tools which require a potentially very long list of files (e.g. training tools) always allow the files to be specified in a script file via the -S option instead of via the command line

```
HInit hmmfile s1 s2 s3 s4 s5 ....
```

```
HInit -S filelist hmmfile
```

```
logfile=physfile[s,e]
```

```
s23-0001-A_000143_000291.plp=/data/plp/complete/s23-0001-A.plp[143,291]
```

```
s23-0001-A_000291_000500.plp=/data/plp/complete/s23-0001-A.plp[291,500]
```

```
s23-0001-A_000500_000889.plp=/data/plp/complete/s23-0001-A.plp[500,889]
```

# Configuration Files

- Configuration files are used for customizing the HTK working environment
- They consist of a list of parameter-values pairs along with an optional prefix which limits the scope of the parameter to a specific module or tool
- The name of a configuration file can be specified explicitly on the command line using the `-C` command

```
HERest ... -C myconfig s1 s2 s3 s4 ...
```



# Configuration Files

- When an explicit configuration file is specified, only those parameters mentioned in that file are actually changed and all other parameters retain their default values
- These defaults are built-in
- User-defined defaults can be set by assigning the name of a default configuration file to the environment variable HCONFIG

```
setenv HCONFIG myconfig  
HERest ... s1 s2 s3 s4 ...
```

```
setenv HCONFIG myconfig  
HERest ... -C xconfig s1 s2 s3 s4 ...
```

# Configuration Files

Configuration Parameters



Config  
File



User  
Defaults



Built-in  
Defaults

# Configuration Files

```
[MODULE:] PARAMETER = VALUE
```

```
# Example config file
```

```
    TARGETKIND = MFCC
```

```
    NUMCHANS   = 20
```

```
    WINDOWSIZE = 250000.0      # ie 25 msecs
```

```
    PREEMCOEF  = 0.97
```

```
    ENORMALISE = T
```

```
HSHELL: TRACE = 02             # octal
```

```
HPARM:  TRACE = 0101
```


# Configuration Files

- -D option can be used to display the configuration file before and after the execution of the tool
- The hash (#) indicates that SAVEBINARY has not been used

```
HTK Configuration Parameters[3]
  Module/Tool      Parameter      Value
#                 SAVEBINARY      TRUE
  HPARM           TARGETRATE      256000.000000
                  TARGETKIND      MFCC_0
```



# Standard Options

- Options consisting of a capital letter are common across all tools
  - -C is used to specify a configuration file name
  - -S is used to specify a script file name
  - -D is used to display configuration settings
  - -A causes the current command line arguments to be printed
  - -V causes version information for the tool and each module used by that tool to be listed
  - -T for trace options
- 

# Error Reporting

- The HShell module provides a standard mechanism for reporting errors and warnings

```
HList: ERROR [+1110]  
IsWave: cannot open file speech.dat
```

- All errors have positive error numbers and always result in the tool terminating. Warnings have negative error numbers and the tool does not terminate
- The first two digits of an error number indicate the module or tool in which the error is located (HList in this case) and the last two digits define the class of error

# Strings and Names

- A name string consists of a single white space delimited word or a quoted string

Notation	Meaning
<code>\\</code>	<code>\</code>
<code>\_</code>	represents a space that will not terminate a string
<code>\'</code>	' (and will not end a quoted string)
<code>\"</code>	" (and will not end a quoted string)
<code>\nnn</code>	the character with octal code <code>\nnn</code>

`"\"QUOTE"`

`\\"QUOTE`


`'\"QUOTE'`

`\042QUOTE`

`"QUOTE.`




# Strings and Names

- A name string consists of a single white space delimited word or a quoted string
  - (a,b,c,d) would be split into 4 distinct name strings a, b, c and d
  - RAWMITFORMAT is set true, each word in a language model definition file consists of a white space delimited string with no special processing being performed
- 






# Memory Management

- Every time that a module or tool wishes to allocate some memory, it does so by calling routines in Hmem
  - To make memory allocation and de-allocation very fast, tools create specific memory allocators for specific objects or groups of objects
  - These memory allocators are divided into a sequence of blocks, and they are organized as either Stacks, M-heaps or C-heaps
- 



# Memory Management

- A Stack constrains the pattern of allocation and de-allocation requests to be made in a last-allocated first-deallocated order but allows objects of any size to be allocated
  - An M-heap allows an arbitrary pattern of allocation and de-allocation requests to be made but all allocated objects must be the same size
  - C-heap uses the underlying operating system and allows arbitrary allocation patterns
  - Most tools provide one or more trace options which show how much memory has been allocated
- 

# Memory Management

- A Stack constrains the pattern of allocation and de-allocation requests to be made in a last-allocated first-deallocated order but allows

```
----- Heap Statistics -----  
nblk=1, siz= 100000*1, used= 32056, alloc= 100000 : Global Stack[S]  
nblk=1, siz=   200*28, used=   100, alloc=   5600 : cellHeap[M]  
nblk=1, siz=  10000*1, used=   3450, alloc=   10000 : mlfHeap[S]  
nblk=2, siz=   7504*1, used=   9216, alloc=   10346 : nameHeap[S]  
-----
```

and allows arbitrary allocation patterns

- Most tools provide one or more trace options which show how much memory has been allocated

## Input / Output via Pipes and Networks

- Most types of file in HTK can be input or output via a pipe instead of directly from or to disk
- The following command will normally list the contents of the speech waveform file spfile

```
HList spfile
```

- If the value of the environment variable HWAVEFILTER is set as follows

```
setenv HWAVEFILTER 'gunzip -c $'
```

- then the effect is to invoke the decompression filter gunzip with its input connected to the file spfile and its output connected to HList via a pipe

# Byte-swapping of HTK data files

- Virtually all HTK tools can read and write data to and from binary files
- The use of binary data format often introduces incompatibilities between different machine architectures due to the different byte ordering conventions used to represent numerical quantities
- All HTK binary data files are written out using big-endian (NONVAX) representation of numerical values
- The default behavior can be altered using the configuration parameters `NATURALREADORDER` and `NATURALWRITEORDER`
- Setting `NATURALREADORDER` to true will instruct the HTK tools to interpret the binary input data in the machine's natural byte order (byte swapping will never take place)



Thank You