NAME

speak – word to voice translator

SYNOPSIS

speak [–**epsv**] [vocabulary [output]]

DESCRIPTION

Speak turns a stream of words into utterances and outputs them to a voice synthesizer, or to a specified output file. It has facilities for maintaining a vocabulary. It receives, from the standard input

- working lines: text of words separated by blanks
- phonetic lines: strings of phonemes for one word preceded and separated by commas. The phonemes may be followed by comma-percent then a 'replacement part' – an ASCII string with no spaces. The phonetic code is given in vsp(VII).
- empty lines
- command lines: beginning with **!.** The following command lines are recognized:

!r file	replace coded vocabulary from file
!w file	write coded vocabulary on file
!p	print parsing for working word
!l	list vocabulary on standard output with phonetics
!c word	copy phonetics from working word to specified word
!d	print phonetics for working word

Each working line replaces its predecessor. Its first word is the 'working word'. Each phonetic line replaces the phonetics stored for the working word. In particular, a phonetic line of comma only deletes the entry for the working word. Each working line, phonetic line or empty line causes the working line to be uttered. The process terminates at the end of input.

Unknown words are pronounced by rules, and failing that, are spelled. Spelling is done by taking each character of the word, prefixing it with *, and looking it up. Unspellable words burp.

Speak is initialized with a coded vocabulary stored in file /usr/lib/speak.m. The vocabulary option substitutes a different file for /usr/lib/speak.m.

A set of single letter options may appear in any order preceded by **–.** Their meanings are:

–e	suppress English steps (4–8) below
–p	suppress pronunciation by rule
–s	suppress spelling
–v	suppress voice output

The steps of pronunciation by rule are:

(1) If there were no lower case letters in the working line, fold all upper case letters to lower.
(2) Fold an initial cap to lower case, and try again.
(3) If word has only one letter, or has no lower case vowels, quit.
(4) If there is a final *s*, strip it.
(5) Replace final –ie by –y.
(6) If any changes have been made, try whole word again.
(7) Locate probable long vowels and capitalize them. Mark probable silent *e*'s.
(8) Put back the *s* stripped in (4), if any.
(9) Place # before and after word.
(10) Prefix word with *%*, and look up longest initial match in the stored table of words; if none, quit.
(11) Use phonemes from the stored phonetic string as pronunciation, and replace the matched stuff by the replacement part of the phonetic string.
(12) If anything remains, go to (10).

Long vowels are located this way in step (7):

(1) A *u* appearing in context [ˆaeiou]u[ˆaeiouwxy][aieouy]. (The notation is just a regular expression à la ed(I).) *(pustUlous)*

(2) One of [aeo] appearing in the context [aeo][ˆaehiouwxy][ie][aou] or in the context [aeo][ˆaehiouwxy]ien is assumed long. The digram *th* behaves as a single letter in this test. *(rAdium, facEtious, quOtient, carpAthian)*

(3) If the first vowel in the word is *i* followed by one of *aou*, it is assumed long. *(Iodine, dIameter, trIumph)*

(4) If the only vowel in the word is final *e*, the vowel is assumed long. *(bE, shE)*

(5) If the only vowels in the word appear in the pattern [aeiouy][ˆaeiouwxy]S, where S is one of the suffixes

–al	–le	–re	–y

then the first vowel is assumed long. *(glObal, tAble, lUcre, lAdy)*

(6) If no suffix was found in (5), as many of these suffixes as possible are isolated from right to left. Stripping stops when *e* has been stripped, nor is *e* stripped before a suffix beginning with *e*. Each suffix is marked by inserting ⏐ just before the first letter, or just after *e* in those suffixes that begin with *e*.

–able	–ably	–e	–ed
–er	–ery	–est	–ful
–ing	–less	–ment	–ness

(care⏐ful⏐ly, maj⏐or, fine⏐ry, state⏐, caree⏐r)

(7) If the word, exclusive of suffixes, ends in *i* or *y*, and contains no earlier vowel, then *i* or *y* is assumed long. *(pY* (from pie), *crY⏐ing, lIe⏐d)*

(8) If the first suffix begins with one of [aeio], then the vowel [aeiouy] in an immediately preceding pattern [ˆaeo][aeiouy][ˆaeiouwxy] is assumed long. The digram *th* behaves as a single letter in this test. *(cAre⏐ful⏐ly, bAthe⏐d, mAj⏐or, pOt⏐able, port⏐able)*

(9) In these exceptional cases no long letter is assumed in the preceding step:
(i) before *g*, if there are any earlier vowels *(postage⏐, stAge⏐, college⏐)*
(ii) *e* is not long before *l (travele⏐d)*

(10) If the first suffix begins with one of [aeio], and the word exclusive of suffixes ends in [aeiouyAEIOUY]th, then digram *th* is capitalized. *(breaTH⏐ing, blITHe⏐ly)*

(11) An attempt is made to recognize silent *e* in the middle of compound words. Such an *e* is marked by a following ⏐, and preceding vowels, other than *e*, are assumed long as in step (8). Silent *e* is marked in the context [bdgmnprst][bdgpt]le[ˆaeioruy⏐]S, where S is any string that contains [aeiouy] but does not contain ⏐ or the end of the word. Silent *e* is also marked in the context [ˆaeiu][aiou][ˆaeiouwxy]e[ˆaeinoruy]S. *(simple⏐ton, fAce⏐guard, cAve⏐man, cavernous)*

FILES

/usr/lib/speak.m

SEE ALSO

vs(VII), vs(IV)

DIAGNOSTICS

'?' for unknown command with **!,** or for unreadable or unwritable vocabulary file

BUGS

Vocabulary overflow is unchecked. Excessively long words cause dumps. Space is not reclaimed from deleted entries.