Language Template B/langt

Purpose

The fundamental definitions needed by the parser and the verb library in order to specify the language of play – that is, the language used for communications between the story file and the player.

§1. Vocabulary.

```
Constant AGAIN1__WD     = 'again';
Constant AGAIN2__WD     = 'g//';
Constant AGAIN3__WD     = 'again';
Constant OOPS1__WD      = 'oops';
Constant OOPS2__WD      = 'o//';
Constant OOPS3__WD      = 'oops';
Constant UNDO1__WD      = 'undo';
Constant UNDO2__WD      = 'undo';
Constant UNDO3__WD      = 'undo';

Constant ALL1__WD       = 'all';
Constant ALL2__WD       = 'each';
Constant ALL3__WD       = 'every';
Constant ALL4__WD       = 'everything';
Constant ALL5__WD       = 'both';
Constant AND1__WD       = 'and';
Constant AND2__WD       = 'and';
Constant AND3__WD       = 'and';
Constant BUT1__WD       = 'but';
Constant BUT2__WD       = 'except';
Constant BUT3__WD       = 'but';
Constant ME1__WD        = 'me';
Constant ME2__WD        = 'myself';
Constant ME3__WD        = 'self';
Constant OF1__WD        = 'of';
Constant OF2__WD        = 'of';
Constant OF3__WD        = 'of';
Constant OF4__WD        = 'of';
Constant OTHER1__WD     = 'another';
Constant OTHER2__WD     = 'other';
Constant OTHER3__WD     = 'other';
Constant THEN1__WD      = 'then';
Constant THEN2__WD      = 'then';
Constant THEN3__WD      = 'then';

Constant NO1__WD        = 'n//';
Constant NO2__WD        = 'no';
Constant NO3__WD        = 'no';
Constant YES1__WD       = 'y//';
Constant YES2__WD       = 'yes';
Constant YES3__WD       = 'yes';

Constant AMUSING__WD    = 'amusing';
```

```
Constant FULLSCORE1__WD = 'fullscore';
Constant FULLSCORE2__WD = 'full';
Constant QUIT1__WD      = 'q//';
Constant QUIT2__WD      = 'quit';
Constant RESTART__WD    = 'restart';
Constant RESTORE__WD    = 'restore';
```

§2. Pronouns.

```
Array LanguagePronouns table
! word        possible GNAs                  connected
!             to follow:                     to:
!             a     i
!             s  p  s  p
!             mfnmfnmfnmfn
    'it'      $$001000111000                   NULL
    'him'     $$100000000000                   NULL
    'her'     $$010000000000                   NULL
    'them'    $$000111000111                   NULL;
```

§3. Descriptors.

```
Array LanguageDescriptors table
! word        possible GNAs   descriptor      connected
!             to follow:      type:           to:
!             a     i
!             s  p  s  p
!             mfnmfnmfnmfn
    'my'      $$111111111111    POSSESS_PK      0
    'this'    $$111111111111    POSSESS_PK      0
    'these'   $$000111000111    POSSESS_PK      0
    'that'    $$111111111111    POSSESS_PK      1
    'those'   $$000111000111    POSSESS_PK      1
    'his'     $$111111111111    POSSESS_PK      'him'
    'her'     $$111111111111    POSSESS_PK      'her'
    'their'   $$111111111111    POSSESS_PK      'them'
    'its'     $$111111111111    POSSESS_PK      'it'
    'the'     $$111111111111    DEFART_PK       NULL
    'a//'     $$111000111000    INDEFART_PK     NULL
    'an'      $$111000111000    INDEFART_PK     NULL
    'some'    $$000111000111    INDEFART_PK     NULL
    'lit'     $$111111111111    light           NULL
    'lighted' $$111111111111    light           NULL
    'unlit'   $$111111111111    (-light)        NULL;
```

§4. Numbers.

```
Array LanguageNumbers table
    'one' 1 'two' 2 'three' 3 'four' 4 'five' 5
    'six' 6 'seven' 7 'eight' 8 'nine' 9 'ten' 10
    'eleven' 11 'twelve' 12 'thirteen' 13 'fourteen' 14 'fifteen' 15
    'sixteen' 16 'seventeen' 17 'eighteen' 18 'nineteen' 19 'twenty' 20
    'twenty-one' 21 'twenty-two' 22 'twenty-three' 23 'twenty-four' 24
    'twenty-five' 25 'twenty-six' 26 'twenty-seven' 27 'twenty-eight' 28
    'twenty-nine' 29 'thirty' 30
;
[ LanguageNumber n f;
    if (n == 0)    { print "zero"; rfalse; }
    if (n < 0)     { print "minus "; n = -n; }
#Iftrue (WORDSIZE == 4);
    if (n >= 1000000000) {
        if (f == 1) print ", ";
        print (LanguageNumber) n/1000000000, " million"; n = n%1000000000; f = 1;
    }
    if (n >= 1000000) {
        if (f == 1) print ", ";
        print (LanguageNumber) n/1000000, " million"; n = n%1000000; f = 1;
    }
#Endif;
    if (n >= 1000) {
        if (f == 1) print ", ";
        print (LanguageNumber) n/1000, " thousand"; n = n%1000; f = 1;
    }
    if (n >= 100)  {
        if (f == 1) print ", ";
        print (LanguageNumber) n/100, " hundred"; n = n%100; f = 1;
    }
    if (n == 0) rfalse;
    #Ifdef DIALECT_US;
    if (f == 1) print " ";
    #Ifnot;
    if (f == 1) print " and ";
    #Endif;
    switch (n) {
    1:    print "one";
    2:    print "two";
    3:    print "three";
    4:    print "four";
    5:    print "five";
    6:    print "six";
    7:    print "seven";
    8:    print "eight";
    9:    print "nine";
    10:   print "ten";
    11:   print "eleven";
    12:   print "twelve";
    13:   print "thirteen";
    14:   print "fourteen";
```

```
    15:   print "fifteen";
    16:   print "sixteen";
    17:   print "seventeen";
    18:   print "eighteen";
    19:   print "nineteen";
    20 to 99: switch (n/10) {
        2:  print "twenty";
        3:  print "thirty";
        4:  print "forty";
        5:  print "fifty";
        6:  print "sixty";
        7:  print "seventy";
        8:  print "eighty";
        9:  print "ninety";
        }
        if (n%10 ~= 0) print "-", (LanguageNumber) n%10;
    }
];
```

§5. Time.

```
[ LanguageTimeOfDay hours mins i;
    i = hours%12;
    if (i == 0) i = 12;
    if (i < 10) print " ";
    print i, ":", mins/10, mins%10;
    if ((hours/12) > 0) print " pm"; else print " am";
];
```

§6. Directions.

```
[ LanguageDirection d;
    print (name) d;
];
```

§7. Translation.

```
[ LanguageToInformese; ];
```

§8. Articles.

```
Constant LanguageAnimateGender   = male;
Constant LanguageInanimateGender = neuter;
Constant LanguageContractionForms = 2;     ! English has two:
                                           ! 0 = starting with a consonant
                                           ! 1 = starting with a vowel

[ LanguageContraction text;
    if (text->0 == 'a' or 'e' or 'i' or 'o' or 'u'
                or 'A' or 'E' or 'I' or 'O' or 'U') return 1;
    return 0;
];

Array LanguageArticles -->

!   Contraction form 0:     Contraction form 1:
!   Cdef   Def    Indef     Cdef   Def    Indef
    "The " "the " "a "      "The " "the " "an "          ! Articles 0
    "The " "the " "some "   "The " "the " "some ";       ! Articles 1

                !             a           i
                !             s     p     s     p
                !             m f n m f n m f n m f n

Array LanguageGNAsToArticles --> 0 0 0 1 1 1 0 0 0 1 1 1;
```

§9. Commands. `LanguageVerbLikesAdverb` is called by `PrintCommand` when printing an `UPTO_PE` error or an inference message. Words which are intransitive verbs, i.e., which require a direction name as an adverb ("walk west"), not a noun ("I only understood you as far as wanting to touch the ground"), should cause the routine to return `true`.

`LanguageVerbMayBeName` is called by `NounDomain` when dealing with the player's reply to a "Which do you mean, the short stick or the long stick?" prompt from the parser. If the reply is another verb (for example, LOOK) then then previous ambiguous command is discarded unless it is one of these words which could be both a verb and an adjective in a `name` property.

```
[ LanguageVerb i;
    switch (i) {
    'i//','inv','inventory':
            print "take inventory";
    'l//':   print "look";
    'x//':   print "examine";
    'z//':   print "wait";
    default: rfalse;
    }
    rtrue;
];

[ LanguageVerbLikesAdverb w;
    if (w == 'look' or 'go' or 'push' or 'walk')
        rtrue;
    rfalse;
];

[ LanguageVerbMayBeName w;
    if (w == 'long' or 'short' or 'normal'
                    or 'brief' or 'full' or 'verbose')
```

```
        rtrue;
    rfalse;
];
```

§10. Short Texts.

```
Constant NKEY__TX       = "N = next subject";
Constant PKEY__TX       = "P = previous";
Constant QKEY1__TX      = "  Q = resume game";
Constant QKEY2__TX      = "Q = previous menu";
Constant RKEY__TX       = "RETURN = read subject";

Constant NKEY1__KY      = 'N';
Constant NKEY2__KY      = 'n';
Constant PKEY1__KY      = 'P';
Constant PKEY2__KY      = 'p';
Constant QKEY1__KY      = 'Q';
Constant QKEY2__KY      = 'q';

Constant SCORE__TX      = "Score: ";
Constant MOVES__TX      = "Moves: ";
Constant TIME__TX       = "Time: ";
Global CANTGO__TX     = "You can't go that way.";
Global FORMER__TX     = "your former self";
Global YOURSELF__TX   = "yourself";
Constant YOU__TX        = "You";
Constant DARKNESS__TX   = "Darkness";

Constant THOSET__TX     = "those things";
Constant THAT__TX       = "that";
Constant OR__TX         = " or ";
Constant NOTHING__TX    = "nothing";
Constant NOTHING2__TX    = "Nothing";
Global IS__TX         = " is";
Global ARE__TX        = " are";
Global IS2__TX        = "is ";
Global ARE2__TX       = "are ";
Global IS3__TX        = "is";
Global ARE3__TX       = "are";
Constant AND__TX        = " and ";
#ifdef SERIAL_COMMA;
Constant LISTAND__TX   = ", and ";
Constant LISTAND2__TX  = " and ";
#ifnot;
Constant LISTAND__TX   = " and ";
Constant LISTAND2__TX  = " and ";
#endif; ! SERIAL_COMMA
Constant WHOM__TX       = "whom ";
Constant WHICH__TX      = "which ";
Constant COMMA__TX      = ", ";
```

§11. Printed Inflections.

```
[ ThatorThose obj;      ! Used in the accusative
    if (obj == player)            { print "you"; return; }
    if (obj has pluralname)       { print "those"; return; }
    if (obj has animate) {
        if (obj has female)       { print "her"; return; }
        else
            if (obj hasnt neuter) { print "him"; return; }
    }
    print "that";
];
[ ItorThem obj;
    if (obj == player)            { print "yourself"; return; }
    if (obj has pluralname)       { print "them"; return; }
    if (obj has animate) {
        if (obj has female)       { print "her"; return; }
        else
            if (obj hasnt neuter) { print "him"; return; }
    }
    print "it";
];
[ IsorAre obj;
    if (obj has pluralname || obj == player) print "are"; else print "is";
];
[ HasorHave obj;
    if (obj has pluralname || obj == player) print "have"; else print "has";
];
[ CThatorThose obj;     ! Used in the nominative
    if (obj == player)            { print "You"; return; }
    if (obj has pluralname)       { print "Those"; return; }
    if (obj has animate) {
        if (obj has female)       { print "She"; return; }
        else
            if (obj hasnt neuter) { print "He"; return; }
    }
    print "That";
];
[ CTheyreorThats obj;
    if (obj == player)             { print "You're"; return; }
    if (obj has pluralname)        { print "They're"; return; }
    if (obj has animate) {
        if (obj has female)        { print "She's"; return; }
        else if (obj hasnt neuter) { print "He's"; return; }
    }
    print "That's";
];
[ HisHerTheir o; if (o has pluralname) { print "their"; return; }
    if (o has female) { print "her"; return; }
    if (o has neuter) { print "its"; return; }
    print "his";
];
```

```
[ HimHerItself o; if (o has pluralname) { print "theirselves"; return; }
    if (o has female) { print "herself"; return; }
    if (o has neuter) { print "itself"; return; }
    print "himself";
];
```

§**12. Long Texts.** The messages here are expected eventually to move into I7 tables, where they will be more easily dealt with. But for now, the old-fashioned way:

```
[ LanguageLM n x1 x2;
say__p = 1;
Answer,Ask:
            "There is no reply.";
! Ask:      see Answer
Attack:   "Violence isn't the answer to this one.";
Burn:     "This dangerous act would achieve little.";
Buy:      "Nothing is on sale.";
Climb:    "I don't think much is to be achieved by that.";
Close: switch (n) {
        1:  print_ret (ctheyreorthats) x1, " not something you can close.";
        2:  print_ret (ctheyreorthats) x1, " already closed.";
        3:  "You close ", (the) x1, ".";
        4: print (The) actor, " closes ", (the) x1, ".^";
        5: print (The) x1, " close"; if (x1 hasnt pluralname) print "s";
            print ".^";
    }
Consult: switch (n) {
        1: "You discover nothing of interest in ", (the) x1, ".";
        2: print (The) actor, " looks at ", (the) x1, ".^";
    }
Cut:      "Cutting ", (thatorthose) x1, " up would achieve little.";
Disrobe: switch (n) {
        1:  "You're not wearing ", (thatorthose) x1, ".";
        2:  "You take off ", (the) x1, ".";
        3: print (The) actor, " takes off ", (the) x1, ".^";
    }
Drink:    "There's nothing suitable to drink here.";
Drop: switch (n) {
        1:  if (x1 has pluralname) print (The) x1, " are "; else print (The) x1, " is ";
            "already here.";
        2:  "You haven't got ", (thatorthose) x1, ".";
        3:  print "(first taking ", (the) x1, " off)^"; say__p = 0; return;
        4:  "Dropped.";
        5: "There is no more room on ", (the) x1, ".";
        6: "There is no more room in ", (the) x1, ".";
        7: print (The) actor, " puts down ", (the) x1, ".^";
    }
Eat: switch (n) {
        1:  print_ret (ctheyreorthats) x1, " plainly inedible.";
        2:  "You eat ", (the) x1, ". Not bad.";
        3: print (The) actor, " eats ", (the) x1, ".^";
    }
Enter: switch (n) {
```

```
        1:  print "But you're already ";
            if (x1 has supporter) print "on "; else print "in ";
            print_ret (the) x1, ".";
        2:  if (x1 has pluralname) print "They're"; else print "That's";
            print " not something you can ";
            switch (verb_word) {
            'stand':  "stand on.";
            'sit':    "sit down on.";
            'lie':    "lie down on.";
            default:  "enter.";
            }
        3:  "You can't get into the closed ", (name) x1, ".";
        4:  "You can only get into something free-standing.";
        5:  print "You get ";
            if (x1 has supporter) print "onto "; else print "into ";
            print_ret (the) x1, ".";
        6:  print "(getting ";
            if (x1 has supporter) print "off "; else print "out of ";
            print (the) x1; print ")^"; say__p = 0; return;
        7:  ! say__p = 0;
            if (x1 has supporter) "(getting onto ", (the) x1, ")";
            if (x1 has container) "(getting into ", (the) x1, ")";
            "(entering ", (the) x1, ")";
        8: print (The) actor, " gets into ", (the) x1, ".^";
        9:  print (The) actor, " gets onto ", (the) x1, ".^";
    }
Examine: switch (n) {
        1:  "Darkness, noun.  An absence of light to see by.";
        2:  "You see nothing special about ", (the) x1, ".";
        3:  print (The) x1, " ", (isorare) x1, " currently switched ";
            if (x1 has on) "on."; else "off.";
        4: print (The) actor, " looks closely at ", (the) x1, ".^";
        5: "You see nothing unexpected in that direction.";
    }
Exit: switch (n) {
        1:  "But you aren't in anything at the moment.";
        2:  "You can't get out of the closed ", (name) x1, ".";
        3:  print "You get ";
            if (x1 has supporter) print "off "; else print "out of ";
            print_ret (the) x1, ".";
        4:  print "But you aren't ";
            if (x1 has supporter) print "on "; else print "in ";
            print_ret (the) x1, ".";
        5: print (The) actor, " gets off ", (the) x1, ".^";
        6: print (The) actor, " gets out of ", (the) x1, ".^";
    }
GetOff:   "But you aren't on ", (the) x1, " at the moment.";
Give: switch (n) {
        1:  "You aren't holding ", (the) x1, ".";
        2:  "You juggle ", (the) x1, " for a while, but don't achieve much.";
        3:  print (The) x1;
            if (x1 has pluralname) print " don't"; else print " doesn't";
            " seem interested.";
```

```
        4:  print (The) x1;
            if (x1 has pluralname) print " aren't";
            else print " isn't";
            " able to receive things.";
        5: "You give ", (the) x1, " to ", (the) second, ".";
        6: print (The) actor, " gives ", (the) x1, " to you.^";
        7: print (The) actor, " gives ", (the) x1, " to ", (the) second, ".^";
    }
Go: switch (n) {
        1:  print "You'll have to get ";
            if (x1 has supporter) print "off "; else print "out of ";
            print_ret (the) x1, " first.";
        2:  print_ret (string) CANTGO__TX;   ! "You can't go that way."
        6:  print "You can't, since ", (the) x1;
            if (x1 has pluralname) " lead nowhere."; else " leads nowhere.";
        7: "You'll have to say which compass direction to go in.";
        8: print (The) actor, " goes up";
        9: print (The) actor, " goes down";
        10: print (The) actor, " goes ", (name) x1;
        11: print (The) actor, " arrives from above";
        12: print (The) actor, " arrives from below";
        13: print (The) actor, " arrives from the ", (name) x1;
        14: print (The) actor, " arrives";
        15: print (The) actor, " arrives at ", (the) x1, " from above";
        16: print (The) actor, " arrives at ", (the) x1, " from below";
        17: print (The) actor, " arrives at ", (the) x1, " from the ", (name) x2;
        18: print (The) actor, " goes through ", (the) x1;
        19: print (The) actor, " arrives from ", (the) x1;
        20: print "on ", (the) x1;
        21: print "in ", (the) x1;
        22: print ", pushing ", (the) x1, " in front, and you along too";
        23: print ", pushing ", (the) x1, " in front";
        24: print ", pushing ", (the) x1, " away";
        25: print ", pushing ", (the) x1, " in";
        26: print ", taking you along";
        27: print "(first getting off ", (the) x1, ")^"; say__p = 0; return;
        28: print "(first opening ", (the) x1, ")^"; say__p = 0; return;
    }
Insert: switch (n) {
        1:  "You need to be holding ", (the) x1, " before you can put ", (itorthem) x1,
            " into something else.";
        2:  print_ret (Cthatorthose) x1, " can't contain things.";
        3:  print_ret (The) x1, " ", (isorare) x1, " closed.";
        4:  "You'll need to take ", (itorthem) x1, " off first.";
        5:  "You can't put something inside itself.";
        6:  print "(first taking ", (itorthem) x1, " off)^"; say__p = 0; return;
        7:  "There is no more room in ", (the) x1, ".";
        8:  "Done.";
        9:  "You put ", (the) x1, " into ", (the) second, ".";
    10:  print (The) actor, " puts ", (the) x1, " into ", (the) second, ".^";
    }
Inv: switch (n) {
        1:  "You are carrying nothing.";
```

```
        2:  print "You are carrying";
        3:  print ":^";
        4:  print ".^";
        5: print (The) x1, " looks through ", (HisHerTheir) x1, " possessions.^";
    }
Jump:     "You jump on the spot, fruitlessly.";
Kiss:     "Keep your mind on the game.";
Listen:   "You hear nothing unexpected.";
ListMiscellany: switch (n) {
        1:  print " (providing light)";
        2:  print " (closed)";
        4:  print " (empty)";
        6:  print " (closed and empty)";
        3:  print " (closed and providing light)";
        5:  print " (empty and providing light)";
        7:  #ifdef SERIAL_COMMA;
            print " (closed, empty, and providing light)";
            #ifnot;
            print " (closed, empty and providing light)";
            #endif;
        8:  print " (providing light and being worn";
        9:  print " (providing light";
        10: print " (being worn";
        11: print " (";
        12: print "open";
        13: print "open but empty";
        14: print "closed";
        15: print "closed and locked";
        16: print " and empty";
        17: print " (empty)";
        18: print " containing ";
        19: print " (on ";
        20: print ", on top of ";
        21: print " (in ";
        22: print ", inside ";
    }
LMode1:   " is now in its normal ~brief~ printing mode, which gives long descriptions
            of places never before visited and short descriptions otherwise.";
LMode2:   " is now in its ~verbose~ mode, which always gives long descriptions
            of locations (even if you've been there before).";
LMode3:   " is now in its ~superbrief~ mode, which always gives short descriptions
            of locations (even if you haven't been there before).";
Lock: switch (n) {
        1:  if (x1 has pluralname) print "They don't "; else print "That doesn't ";
            "seem to be something you can lock.";
        2:  print_ret (ctheyreorthats) x1, " locked at the moment.";
        3:  "First you'll have to close ", (the) x1, ".";
        4:  if (x1 has pluralname) print "Those don't "; else print "That doesn't ";
            "seem to fit the lock.";
        5:  "You lock ", (the) x1, ".";
        6: print (The) actor, " locks ", (the) x1, ".^";
    }
Look: switch (n) {
```

```
        1:  print " (on ", (the) x1, ")";
        2:  print " (in ", (the) x1, ")";
        3:  print " (as ", (object) x1, ")";
        4:  print "On ", (the) x1, " ";
            WriteListFrom(child(x1),
            ENGLISH_BIT+RECURSE_BIT+PARTINV_BIT+TERSE_BIT+CONCEAL_BIT+ISARE_BIT);
            ".";
        5,6:
            if (x1 ~= location) {
                if (x1 has supporter) print "On "; else print "In ";
                print (the) x1, " you";
            }
            else print "You";
            print " can ";
            if (n == 5) print "also ";
            print "see ";
            WriteListFrom(child(x1),
            ENGLISH_BIT+RECURSE_BIT+PARTINV_BIT+TERSE_BIT+CONCEAL_BIT+WORKFLAG_BIT);
            if (x1 ~= location) "."; else " here.";
        7:  "You see nothing unexpected in that direction.";
        8:  if (x1 has supporter) print " (on "; else print " (in ";
            print (the) x1, ")";
        9: print (The) actor, " looks around.^";
    }
LookUnder: switch (n) {
        1:  "But it's dark.";
        2:  "You find nothing of interest.";
        3: print (The) actor, " looks under ", (the) x1, ".^";
    }
Mild:     "Quite.";
Miscellany: switch (n) {
        1:  "(considering the first sixteen objects only)^";
        2:  "Nothing to do!";
        3:  print " You have died ";
        4:  print " You have won ";
        5:  print "^Would you like to RESTART, RESTORE a saved game";
            #Ifdef DEATH_MENTION_UNDO;
            print ", UNDO your last move";
            #Endif;
            #ifdef SERIAL_COMMA;
            print ",";
            #endif;
            " or QUIT?";
        6:  "[Your interpreter does not provide ~undo~.  Sorry!]";
            #Ifdef TARGET_ZCODE;
        7:  "~Undo~ failed.  [Not all interpreters provide it.]";
            #Ifnot; ! TARGET_GLULX
        7:  "[You cannot ~undo~ any further.]";
            #Endif; ! TARGET_
        8:  "Please give one of the answers above.";
        9:  "It is now pitch dark in here!";
        10: "I beg your pardon?";
        11: "[You can't ~undo~ what hasn't been done!]";
```

```
12: "[Can't ~undo~ twice in succession. Sorry!]";
13: "[Previous turn undone.]";
14: "Sorry, that can't be corrected.";
15: "Think nothing of it.";
16: "~Oops~ can only correct a single word.";
17: "It is pitch dark, and you can't see a thing.";
18: print "yourself";
19: "As good-looking as ever.";
20: "To repeat a command like ~frog, jump~, just say ~again~, not ~frog, again~.";
21: "You can hardly repeat that.";
22: "You can't begin with a comma.";
23: "You seem to want to talk to someone, but I can't see whom.";
24: "You can't talk to ", (the) x1, ".";
25: "To talk to someone, try ~someone, hello~ or some such.";
26: "(first taking ", (the) x1, ")";
27: "I didn't understand that sentence.";
28: print "I only understood you as far as wanting to ";
29: "I didn't understand that number.";
30: "You can't see any such thing.";
31: "You seem to have said too little!";
32: "You aren't holding that!";
33: "You can't use multiple objects with that verb.";
34: "You can only use multiple objects once on a line.";
35: "I'm not sure what ~", (address) pronoun_word, "~ refers to.";
36: "You excepted something not included anyway!";
37: "You can only do that to something animate.";
    #Ifdef DIALECT_US;
38: "That's not a verb I recognize.";
    #Ifnot;
38: "That's not a verb I recognise.";
    #Endif;
39: "That's not something you need to refer to in the course of this game.";
40: "You can't see ~", (address) pronoun_word, "~ (", (the) pronoun_obj,
    ") at the moment.";
41: "I didn't understand the way that finished.";
42: if (x1 == 0) print "None"; else print "Only ", (number) x1;
    print " of those ";
    if (x1 == 1) print "is"; else print "are";
    " available.";
43: "Nothing to do!";
44: "There are none at all available!";
45: print "Who do you mean, ";
46: print "Which do you mean, ";
47: "Sorry, you can only have one item here. Which exactly?";
48: print "Whom do you want";
    if (actor ~= player) print " ", (the) actor;
    print " to "; PrintCommand(); print "?^";
49: print "What do you want";
    if (actor ~= player) print " ", (the) actor;
    print " to "; PrintCommand(); print "?^";
50: print "Your score has just gone ";
    if (x1 > 0) print "up"; else { x1 = -x1; print "down"; }
    print " by ", (number) x1, " point";
```

```
            if (x1 > 1) print "s";
        51: "(Since something dramatic has happened, your list of commands has been cut short.)";
        52: "^Type a number from 1 to ", x1, ", 0 to redisplay or press ENTER.";
        53: "^[Please press SPACE.]";
        54: "[Comment recorded.]";
        55: "[Comment NOT recorded.]";
        56: print ".^";
        57: print "?^";
        58: print (The) actor, " ", (IsOrAre) actor, " unable to do that.^";
        59: "You must supply a noun.";
        60: "You may not supply a noun.";
        61: "You must name an object.";
        62: "You may not name an object.";
        63: "You must name a second object.";
        64: "You may not name a second object.";
        65: "You must supply a second noun.";
        66: "You may not supply a second noun.";
        67: "You must name something more substantial.";
        68: print "(", (The) actor, " first taking ", (the) x1, ")^";
        69: "(first taking ", (the) x1, ")";
        70: "The use of UNDO is forbidden in this game.";
        71: print (string) DARKNESS__TX;
        72: print (The) x1;
            if (x1 has pluralname) print " have"; else print " has";
            " better things to do.";
        73: "That noun did not make sense in this context.";
        74: print "[That command asks to do something outside of play, so it can
            only make sense from you to me. ", (The) x1, " cannot be asked to do this.]^";
        75:  print " The End ";
    }
No,Yes:   "That was a rhetorical question.";
NotifyOff:
            "Score notification off.";
NotifyOn: "Score notification on.";
Open: switch (n) {
        1:  print_ret (ctheyreorthats) x1, " not something you can open.";
        2:  if (x1 has pluralname) print "They seem "; else print "It seems ";
            "to be locked.";
        3:  print_ret (ctheyreorthats) x1, " already open.";
        4:  print "You open ", (the) x1, ", revealing ";
            if (WriteListFrom(child(x1), ENGLISH_BIT+TERSE_BIT+CONCEAL_BIT) == 0) "nothing.";
            ".";
        5:  "You open ", (the) x1, ".";
        6: print (The) actor, " opens ", (the) x1, ".^";
        7: print (The) x1, " open";
            if (x1 hasnt pluralname) print "s";
            print ".^";
    }
Pronouns: switch (n) {
        1:  print "At the moment, ";
        2:  print "means ";
        3:  print "is unset";
        4:  "no pronouns are known to the game.";
```

```
        5:  ".";
    }
Pull,Push,Turn: switch (n) {
        1:  if (x1 has pluralname) print "Those are "; else print "It is ";
            "fixed in place.";
        2:  "You are unable to.";
        3:  "Nothing obvious happens.";
        4:  "That would be less than courteous.";
        5: print (The) actor, " pulls ", (the) x1, ".^";
        6: print (The) actor, " pushes ", (the) x1, ".^";
        7: print (The) actor, " turns ", (the) x1, ".^";
    }
! Push: see Pull
PushDir: switch (n) {
        1:  print (The) x1, " cannot be pushed from place to place.^";
        2:  "That's not a direction.";
        3:  "Not that way you can't.";
    }
PutOn: switch (n) {
        1:  "You need to be holding ", (the) x1, " before you can put ",
                (itorthem) x1, " on top of something else.";
        2:  "You can't put something on top of itself.";
        3:  "Putting things on ", (the) x1, " would achieve nothing.";
        4:  "You lack the dexterity.";
        5:  print "(first taking ", (itorthem) x1, " off)^"; say__p = 0; return;
        6:  "There is no more room on ", (the) x1, ".";
        7:  "Done.";
        8:  "You put ", (the) x1, " on ", (the) second, ".";
        9:  print (The) actor, " puts ", (the) x1, " on ", (the) second, ".^";
    }
Quit: switch (n) {
        1:  print "Please answer yes or no.";
        2:  print "Are you sure you want to quit? ";
    }
Remove: switch (n) {
        1:  if (x1 has pluralname) print "They are"; else print "It is";
            " unfortunately closed.";
        2:  if (x1 has pluralname) print "But they aren't"; else print "But it isn't";
            " there now.";
        3:  "Removed.";
    }
Restart: switch (n) {
        1:  print "Are you sure you want to restart? ";
        2:  "Failed.";
    }
Restore: switch (n) {
        1:  "Restore failed.";
        2:  "Ok.";
    }
Rub:      "You achieve nothing by this.";
Save: switch (n) {
        1:  "Save failed.";
        2:  "Ok.";
```

```
    }
Score: switch (n) {
        1:  if (deadflag) print "In that game you scored "; else print "You have so far scored ";
            print score, " out of a possible ", MAX_SCORE, ", in ", turns, " turn";
            if (turns ~= 1) print "s";
            return;
        2:  "There is no score in this story.";
        3: print ", earning you the rank of ";
    }
ScriptOff: switch (n) {
        1:  "Transcripting is already off.";
        2:  "^End of transcript.";
        3:  "Attempt to end transcript failed.";
    }
ScriptOn: switch (n) {
        1:  "Transcripting is already on.";
        2:  "Start of a transcript of";
        3:  "Attempt to begin transcript failed.";
    }
Search: switch (n) {
        1:  "But it's dark.";
        2:  "There is nothing on ", (the) x1, ".";
        3:  print "On ", (the) x1, " ";
            WriteListFrom(child(x1), ENGLISH_BIT+TERSE_BIT+CONCEAL_BIT+ISARE_BIT);
            ".";
        4:  "You find nothing of interest.";
        5:  "You can't see inside, since ", (the) x1, " ", (isorare) x1, " closed.";
        6:  print_ret (The) x1, " ", (isorare) x1, " empty.";
        7:  print "In ", (the) x1, " ";
            WriteListFrom(child(x1), ENGLISH_BIT+TERSE_BIT+CONCEAL_BIT+ISARE_BIT);
            ".";
        8: print (The) actor, " searches ", (the) x1, ".^";
    }
SetTo:    "No, you can't set ", (thatorthose) x1, " to anything.";
Show: switch (n) {
        1:  "You aren't holding ", (the) x1, ".";
        2:  print_ret (The) x1, " ", (isorare) x1, " unimpressed.";
    }
Sing:     "Your singing is abominable.";
Sleep:    "You aren't feeling especially drowsy.";
Smell:    "You smell nothing unexpected.";
            #Ifdef DIALECT_US;
Sorry:    "Oh, don't apologize.";
            #Ifnot;
Sorry:    "Oh, don't apologise.";
            #Endif;
Squeeze: switch (n) {
        1:  "Keep your hands to yourself.";
        2:  "You achieve nothing by this.";
        3: print (The) actor, " squeezes ", (the) x1, ".^";
    }
Strong:   "Real adventurers do not use such language.";
Swing:    "There's nothing sensible to swing here.";
```

```
SwitchOff: switch (n) {
        1:  print_ret (ctheyreorthats) x1, " not something you can switch.";
        2:  print_ret (ctheyreorthats) x1, " already off.";
        3:  "You switch ", (the) x1, " off.";
        4: print (The) actor, " switches ", (the) x1, " off.^";
    }
SwitchOn: switch (n) {
        1:  print_ret (ctheyreorthats) x1, " not something you can switch.";
        2:  print_ret (ctheyreorthats) x1, " already on.";
        3:  "You switch ", (the) x1, " on.";
        4: print (The) actor, " switches ", (the) x1, " on.^";
    }
Take: switch (n) {
        1:  "Taken.";
        2:  "You are always self-possessed.";
        3:  "I don't suppose ", (the) x1, " would care for that.";
        4:  print "You'd have to get ";
            if (x1 has supporter) print "off "; else print "out of ";
            print_ret (the) x1, " first.";
        5:  "You already have ", (thatorthose) x1, ".";
        6:  if (noun has pluralname) print "Those seem "; else print "That seems ";
            "to belong to ", (the) x1, ".";
        7:  if (noun has pluralname) print "Those seem "; else print "That seems ";
            "to be a part of ", (the) x1, ".";
        8:  print_ret (Cthatorthose) x1, " ", (isorare) x1,
            "n't available.";
        9:  print_ret (The) x1, " ", (isorare) x1, "n't open.";
        10: if (x1 has pluralname) print "They're "; else print "That's ";
            "hardly portable.";
        11: if (x1 has pluralname) print "They're "; else print "That's ";
            "fixed in place.";
        12: "You're carrying too many things already.";
        13: print "(putting ", (the) x1, " into ", (the) x2,
            " to make room)^"; say__p = 0; return;
        14: "You can't reach into ", (the) x1, ".";
        15: "You cannot carry ", (the) x1, ".";
        16: print (The) actor, " picks up ", (the) x1, ".^";
    }
Taste:    "You taste nothing unexpected.";
Tell: switch (n) {
        1:  "You talk to yourself a while.";
        2:  "This provokes no reaction.";
    }
Think:    "What a good idea.";
ThrowAt: switch (n) {
        1:  "Futile.";
        2:  "You lack the nerve when it comes to the crucial moment.";
    }
Tie: "You would achieve nothing by this.";
Touch: switch (n) {
        1:  "Keep your hands to yourself!";
        2:  "You feel nothing unexpected.";
        3:  "If you think that'll help.";
```

```
        4: print (The) actor, " touches ", (himheritself) x1, ".^";
        5: print (The) actor, " touches you.^";
        6: print (The) actor, " touches ", (the) x1, ".^";
    }
! Turn: see Pull.
Unlock:  switch (n) {
        1:  if (x1 has pluralname) print "They don't "; else print "That doesn't ";
            "seem to be something you can unlock.";
        2:  print_ret (ctheyreorthats) x1, " unlocked at the moment.";
        3:  if (x1 has pluralname) print "Those don't "; else print "That doesn't ";
            "seem to fit the lock.";
        4:  "You unlock ", (the) x1, ".";
        5: print (The) actor, " unlocks ", (the) x1, ".^";
    }
Verify: switch (n) {
        1:  "The game file has verified as intact.";
        2:  "The game file did not verify as intact, and may be corrupt.";
    }
Wait: switch (n) {
        1:  "Time passes.";
        2: print (The) actor, " waits.^";
    }
Wake:     "The dreadful truth is, this is not a dream.";
WakeOther:"That seems unnecessary.";
Wave: switch (n) {
        1:  "But you aren't holding ", (thatorthose) x1, ".";
        2:  "You look ridiculous waving ", (the) x1, ".";
        3: print (The) actor, " waves ", (the) x1, ".^";
    }
WaveHands:"You wave, feeling foolish.";
Wear: switch (n) {
        1:  "You can't wear ", (thatorthose) x1, "!";
        2:  "You're not holding ", (thatorthose) x1, "!";
        3:  "You're already wearing ", (thatorthose) x1, "!";
        4:  "You put on ", (the) x1, ".";
        5: print (The) actor, " puts on ", (the) x1, ".^";
    }
! Yes:  see No.
];
```

§**13. Printing Mechanism.** The following routine produces a "library message" – though the library is no more, the terminology lives on.

The `L__M` routine is designed to reach up into I7 to offer it a chance to intervene, but then go back to the I6 method if it doesn't. The Standard Rules ordinarily define these three routines as stubs which always return false, so by default there's no intervention. (This is the hook for the new model of library messages which will be introduced in future builds.)

We divide into three cases because `##Miscellany` and `##ListMiscellany` are fake actions, not actions, in I6. This means it would not be type-safe to store them in I7 variables whose kind of value is "action name", and that would make any single I7 routine handling all three kinds of message quite difficult to write.

```
[ L__M act n x1 x2 rv flag;
    @push sw__var;
    sw__var = act;
    if (n == 0) n = 1;
    @push action;
    lm_act = act;
    lm_n = n;
    lm_o = x1;
    lm_o2 = x2;
    switch (act) {
        ##Miscellany: rv = (+ whether or not intervened in miscellaneous message +);
        ##ListMiscellany: rv = (+ whether or not intervened in miscellaneous list message +);
        default: rv = (+ whether or not intervened in action message +);
    }
    action = sw__var;
    if (rv == false) rv = RunRoutines(LibraryMessages, before);
    @pull action;
    if (rv == false) LanguageLM(n, x1, x2);
    @pull sw__var;
];
```