"READ ROUTINE"

```
Smalltalk define ☞Readerstuff as ☞Readerstuff ← SymbolTable new.!
Readerstuff insertall: ☞(typetable scantable dot
    separ letter digit notapos notcomnt notcr).!

Class new title: 'Reader';
    fields: 'source sink token nextchar typetbl scantbl';
    sharing: 'Readerstuff'~
Reader understands: 'of: source
    [typetbl ← typetable. scantbl ← scantable.
    sink ← (Vector new: 10) asStream. token ← Stream default.
    self step]'!
Reader understands: 'step
    [nextchar ← source next]'!    "nextchar avoids backing up"
Reader understands: 'read | x
    [while§ nextchar do§
        [x ← typetbl·(nextchar+1).
        x=1⇒ [self scan: separ];          "separators"
        =2⇒ [sink next ← (self scan: letter+digit) unique];"identifiers"
        =3⇒ [sink next ← (self scan: 0) unique]; "1-char tokens"
        =4⇒ [sink next ← self rdnum. dot⇒[sink next← ☞.]];"Numbers"
        =5⇒ [sink next ← self rdstr];          "Strings"
        =6⇒ [self step. sink next ← self subread]; "sub-Vectors"
        =7⇒ [self step. ↑sink contents];"close-paren"
        =8⇒ [self rdcom];          "comments"
        =9⇒ [self scan: notcr];          "↑Z format runs"
        =10⇒ [↑sink contents];          "null and DOIT"
        ]
    ↑sink contents]'!
Reader understands: 'scan: mask | x
    [mask=0 ⇒[x ← String new: 1. x·1 ← nextchar. self step. ↑x]
    token reset.
    while§ nextchar do§
        [(mask land: scantbl·(nextchar+1))=0⇒ [↑token contents]
        token next ← nextchar. nextchar ← source next]
    ↑token contents]'!
Reader understands: 'subread | a b
    [a ← sink. sink ← (Vector new: 10) asStream.
    b ← self read. sink ← a. ↑b]'!
Reader understands: 'rdnum | sign val d
    [sign ← [nextchar=025⇒ [self step. ~1] 1]."sign"
    d ← [nextchar=060⇒ [8] 10].          "base"
    val ← self mknum: (self scan: digit) base: d."integer part"
    dot ← false.
    nextchar=056⇒    "check for decimal point"
        [self step.
        nextchar isdigit@false⇒
            [dot ← true. ↑sign*val]    "was <Integer> . "
        d ← self scan: digit.
        val ← (self mknum: d base: 10)asFloat / (10.0 ipow: d length) + val.
        nextchar=0145⇒          "check for e<exponent>"
            [self step. ↑val*(10.0 ipow: self rdnum)*sign]
        ↑val*sign]
    ↑sign*val]'!
Reader understands: 'mknum: str base: base | val c
    [val ← [str length>4⇒ [0.0] 0].
    for§ c from: str do§
        [val ← val*base + (c-060)]
    ↑val]'!
Reader understands: 'rdstr | a
    [self step. a ← self scan: notapos.
    nextchar @ false⇒ [user notify: "Unmatched String quote"]
    self step.
    nextchar=047⇒"imbedded String-quote"
        [token next ← 047. a← token contents
```

```
        ⇑a + self rdstr]
    ⇑a]'!
Reader understands: 'rdcom
    [self step.  self scan: notcomnt.
    nextchar@false⇒ [user notify: "Unmatched comment quote"]
    self step]'!

☞Readerinit ← function (type bit asc1 asc2 typetbl scantbl i)  "init read tables in-line"
    (☞type ← :. ☞bit ← 8. (bit=0⇒() ☞bit ← Readerstuff·bit).
    ☞asc1 ← :. ☞asc2 ← (☜to⇒(:) asc1).
    ☞typetbl ← Readerstuff·☞typetable. ☞scantbl ← Readerstuff·☞scantable.
    (type=0⇒() for i← asc1+1 to asc2+1 do (typetbl[i]  ← type)).
    for i← asc1+1 to asc2+1 do
        (scantbl[i] ← scantbl[i] ⊟bit)
    Readerstuff·☞typetable set typetbl. Readerstuff·☞scantable set scantbl.
    ⇑nil).!
Readerstuff insertall: ☞(separ letter digit notapos notcr notcomnt)
    with: ☞(1 2 4 8 16 32).!
Readerstuff define ☞typetable as string 256.!
Readerstuff·☞typetable[1 to 256] ← all 3.!  "1-char tokens"
Readerstuff define ☞scantable as string 256.!
Readerstuff·☞scantable[1 to 256] ← all 8+16+32.!
Readerinit 0  notcr  015.  "cr"
Readerinit 1  separ  011!  "separators: tab, "
Readerinit 1  separ  012!  " LF, "
Readerinit 1  separ  014!  " FF, "
Readerinit 1  separ  015!  " CR, "
Readerinit 1  separ  040!  " blank "
Readerinit 2  letter  0101 to 0132!  "letters"
Readerinit 2  letter  0141 to 0172!"lower-case"
Readerinit 3  letter  072!  "colon"
Readerinit 3  letter  03!       "open colon"
Readerinit 4  digit  060 to 071!  "digits"
Readerinit 4  0  025!  "high-minus"
Readerinit 5  notapos  047!  "String-quote"
Readerinit 6  0  050!  "left-paren"
Readerinit 7  0  051!  "close-paren"
Readerinit 8  notcomnt  017!  "comments"
Readerinit 9  0  032!  "↑Z format runs"
Readerinit 10  0  036!  "null and DOIT"
```

"BYTE COMPILER"

```
Smalltalk define ☞Compilerstuff as ☞Compilerstuff ← SymbolTable new.!
Compilerstuff insertall: ☞(cdict opdict initcdict initopdict selector nargs ntemps maxtemp
        environment literals precode stack maxstack canoptpop).!
Compilerstuff define ☞trace as false!
Compilerstuff insertall: ☞(instcode tempcode litcode liticode areccode constcode selfcode nilcode
        smashpcode smashcode popcode returncode endcode curcode supercode
        shortjmp shortbfp jmpcode bfpcode minopcode opcode)
    with: ☞(0 16 32 64 112 120 113 125
        128 129 130 131 132 133 134
        144 152 160 168 176 208).!

Class new title: 'Compiler';
    fields: 'source code cascading stackused fixups';
    sharing: 'Compilerstuff'~
Compiler understands: 'compile: b in: class | a c i t
    [self init: class.  a ← b asVector asStream.
    self compilepattern: a.              "pattern and decls"
    self compileblock: a. self push.    "Smalltalk code"
    [self nofixups⇒[] "no redundant return"
        [stack=maxstack⇒[self popopt]].
        code next← selfcode. code next← returncode].
    [
    c ← [a ◂☞primitive: ⇒[a next] 0]."primitive: <integer> clause"
    a← self qcodeck: c ⇒ []     "ckeck for rd/wrt accessors"
    a ← Stream default.                          "now make up code object"
    a next← 0; next← c; next← maxtemp+maxstack.    "header"
    a next← nargs; next← maxtemp; next← 6+(2*literals length).
    for° i from: literals do°                                "literals"
        [a next← i PTR lshift: ¯8; next← i PTR land: 0377]
    a append: precode contents; append: code contents.       "code body"
    ].
    class install: selector
        method: (code←a contents) literals: [literals length=0⇒[literals] nil]
        code: b backpointers: nil.
    ⇑selector]'!
Compiler understands: 'qcodeck: t | a b i "fast read/write accessors"
    [t≠0⇒[⇑false] code loc≠2⇒[⇑false]
    a ← Stream default. a next← 0.
    code·2≠returncode⇒ [⇑false]
    code·1≠selfcode⇒
        [code·1<tempcode⇒
            [precode empty⇒
                [a next← 40; next← 0; next← 0; next← code·1. ⇑a]"quick ⇑inst var"
            ⇑false]
        ⇑false]
    precode empty⇒[a next← 1. ⇑a]    "quick ⇑self"
    b ← precode contents. nargs≠b length/3⇒[⇑false]
    a next← 41; next← 0; next← b length/3.
    for° i from: (3 to: b length by: 3) do°
        [b·i<tempcode⇒[a next← b·i] ⇑false]      "write inst vars"
    ⇑a]'!
Compiler understands: 'init: class | a i
    [environment ← class environment.
    [initcdict@nil⇒     "initial symbol defs"
        [initcdict ← Dictionary new init: 16.
        initcdict insertall: ☞(self thisContext super nil false true)
            with: ☞(113 133 134 125 126 127).
        initopdict ← Dictionary new init: 64.
        initopdict insertall: ☞(+ - < > ≤ ≥ = ≠
                * / \ | min: max: land: lor:
                · ⇒("'←"atomize) next ⇒("next←"atomize) length @
                class and: or: new new: to: oneToMeAsStream asStream)
            with: ☞(176 177 178 179 180 181 182 183
                184 185 186 187 188 189 190 191
                192 193 194 195 196 197
                200 201 202 203 204 205 206 207)] ].
    cdict ← Dictionary new copyfrom: initcdict.
    opdict ← Dictionary new copyfrom: initopdict.
```

```
    a ← class instvars. for⁸ i to: a length do⁸
        [cdict insert: a'i with: instcode+i-1].
    stack← ntemps← maxtemp← 0. maxstack← 1.
    precode ← Stream default.
    literals ← Vector new: 0]'!

Compiler understands: 'compilepattern: a | c
    [ntemps ← 0. selector ← a peek.
    [selector iskeyword⇒        "selector and args"
        [c ← nullString.
        until⁸ [a end or: a peek iskeyword @ false] do⁸        "keywords"
            [c ← c+ a next.
            self compilearg: a next]
        selector ← c unique]
    a next.
    selector isinfix⇒        "infix"
        [self compilearg: a next]
    ].
    [a ◀☞← ⇒                "possible ← phrase"
        [selector ← (selector + "←") unique.
        self compilearg: a next]
    ].
    nargs ← ntemps.
    a ◀☞| ⇒                "further temp declarations"
        [until⁸ [a end or: ☞[=a peek] do⁸
            [self compiletemp: a next]]]'!
Compiler understands: 'compilearg: a | b
    [b ← self newtemp.
    cdict has: a⇒
        [precode next← b.        "compile assignments"
        precode next← smashpcode.    "for non-temp args"
        precode next← cdict'a.  maxstack ← 1]
    cdict insert: a with: b]'!
Compiler understands: 'compiletemp: a | b
    [b ← self newtemp.
    cdict has: a⇒
        [user notify: "temp name used elsewhere: "+a asString]
    cdict insert: a with: b]'!
Compiler understands: 'newtemp
    [maxtemp ← maxtemp max: (ntemps ← ntemps+1).
    ⇑tempcode + ntemps-1]'!

Compiler understands: 'compileblock: source | a b c inblock
    [cascading ← false.
    code ← Stream default. fixups ← (Vector new: 0) asStream.
    source end⇒[⇑nullString] a ← stack.
    [source ◀☞[⇒ []
        user notify: "[ missing before: "+source peek asString].
    while⁸ inblock do⁸
        [source end⇒[user notify: "unbalanced brackets"]
        source ◀☞] ⇒[code next← nilcode. self push. inblock ← false]
        source ◀☞⇑ ⇒[code append: (self compileexpr: source). self push.
            code next← returncode. source ◀☞. . source ◀☞] ⇒[inblock ← false]
            user show:  "junk following return stmt".
            inblock ← false]
        self controlstmt ⇒[self uncascade. source ◀☞.]
        canoptpop ← true.
        code append: [cascading⇒[self compilecascade: source]
            self compileexpr: source]. self push.
        source ◀☞] ⇒[inblock ← false]
        source ◀☞⇒ ⇒[self pop. c ← Compiler new.
            b ← c compileblock: source.
                [source ◀☞; ⇒[self cascade]
                self uncascade. source ◀☞.].
            c nofixups⇒
                [code append: (self encodejmp: bfpcode by: b length); append: b]
            fixups next← code contents; next← b. code reset]
        [source ◀☞; ⇒[self cascade]
        self uncascade.  source ◀☞.⇒[]
         user show: "per missing before: "+source peek asString ].
        self pop. self popopt]        "period pops stack"
```

```
    self pop. stack≠a⇒[user notify: "unbalanced stack"]
    fixups empty⇒ [⇑code contents] canoptpop←false.
    self fixup: code contents and: fixups contents.
    ⇑code contents]'!
Compiler understands: 'popopt | t      "append pop and optimize"
    [2>(t←code loc)⇒[t=1⇒[code skip: ¯1]]
    canoptpop@false⇒ [code next← popcode]
    code˙t=nilcode⇒              "jmpI-nil-pop -> pop"
        [[code˙(t-1)=shortjmp⇒[code skip: ¯2]]. code next← popcode]
    code˙(t-1)=smashcode⇒
        [code˙(t-1) ← smashpcode]"sto-var-pop -> stopop var"
    code next← popcode]'!
Compiler understands: 'cascade | t
    [cascading⇒[]      "cascading is either false"
    (t ← code˙code loc)<minopcode⇒
        [user notify: "improper cascading [...;]."]
    code˙(code loc-1)<arecode⇒
        [cascading ← code˙(code loc-1)]  "or = temp code for receiver"
    code skip: ¯1. code next← smashcode.      "alloc temp if not simple"
    code next← cascading ← self newtemp. code next← t]'!
Compiler understands: 'uncascade
    [cascading⇒[cascading<areccode⇒[cascading ← false]
        ntemps ← ntemps-1. cascading ← false]]'!
Compiler understands: 'nofixups"!true if no jmpout needed"
    [fixups empty⇒          "...meaning no internal branches"
        [code loc=0⇒[⇑false]  ⇑code˙code loc=returncode]"AND ends with ⇑"
    ⇑false]'!
Compiler understands: 'fixup: a and: b | c i t
        "b˙odd=main code, c˙odd ← bfp around consequents,
        b˙even=consequents, c˙even ← jmp to end"
    [c ← Vector new: b length. t ← a length.
    for⁸ i from: (b length-1 to: 1 by: ¯2) do⁸      "compute the jumps"
        [c˙(i+1) ← self encodejmp: jmpcode by: t.
        c˙i ← self encodejmp: bfpcode by: ((b˙(i+1)) length+(c˙(i+1)) length).
        t ← t+(b˙i) length+(b˙(i+1)) length+(c˙i) length+(c˙(i+1)) length]
    code reset. for⁸ i to: b length do⁸      "collate all back into code"
        [code append: b˙i; append: c˙i]
    code append: a]'!
Compiler understands: 'controlstmt
    [source ☞for⁸ ⇒[self compilefor]
    source ☞until⁸ ⇒[self compileuntil: true]
    source ☞while⁸ ⇒[self compileuntil: false]
    ⇑false]'!
Compiler understands: 'compilefactor: s [⇑self compilephrase: 1 from: s]'!
Compiler understands: 'compileterm: s [⇑self compilephrase: 2 from: s]'!
Compiler understands: 'compileexpr: s [⇑self compilephrase: 3 from: s]'!
Compiler understands: 'compilecascade: s [⇑self compilephrase: 4 from: s]'!
Compiler understands: 'compilephrase: level from: input | a b c i t stk more
    [a ← Stream default.
        [level=4⇒ [a next← cascading. level←3]
        self track⁸ (self compileprimary: input into: a level: level)⇒["no assign"]
        ⇑(self compileexpr: input) + a contents].
    stk ← stack. b ← (Vector new: 4) asStream.   "arg stack"
    b next← a; next← stackused.  a ← Stream default.          "operators"
    for⁸ i to: level do⁸
        [more ← true.  while⁸ more do⁸
            [input end⇒[more ← false] c←input peek.
            (c is: UniqueString) @ false⇒
                    [user notify: "unexpected: "+ c asString]
                [i=1⇒[c isinfix⇒[more ← false]
                    c iskeyword⇒[more ← false]  c ← input next];
                =2⇒[c iskeyword⇒[more ← false]
                    ☞( . ; ⇒ [ ] ← ⇑) has: c⇒[more ← false]
                    c ← input next.
                    b next← self track⁸ (self compilefactor: input).
                    b next← stackused];
                =3⇒[☞(for⁸ until⁸ while⁸) has: c⇒[more ← false]
                    while⁸ [input end⇒[false] (c←input peek) iskeyword] do⁸
                        [t ← [t @ nil⇒[input next] t + input next].
```

```
                    a append: [c isuneval⇒
                            [self remote: (self compileterm: input)]
                        self compileterm: input].
                    self push]
                t@nil⇒[more ← false] c ← t unique]].
            more@false⇒[]
                [level=3⇒[input ☞← ⇒          "check here for ← clause"
                    [c←(c+"←")unique.
                    [i=2⇒[self exstack: b pop. a append: b pop. self push]].
                    a append: (self compileexpr: input). self push]]].
            (b`1) next← self encodeop: c. i=3⇒[more ← false]]]
        b`1 ← (b`1) contents. until: b empty do:
            [self exstack: b pop. a append: b pop. self push]
        stack ← stk. ⇑a contents]'!
Compiler understands: 'compileprimary: input into: a level: n | c
    [☞[ = input peek ⇒[c ← Compiler new. a append: (c compileblock: input)]
    c ← input next.
    c is: Vector⇒[a append: (self compileexpr: (c← Stream new of: c)).
        c end⇒[] user notify: "extra suff in parens: "+ a asString]
    c ← [☞☞=c⇒[self encodelit: input next] self encoderef: c].
    [n=3⇒[input ☞← ⇒              "expr-level call checks for ← "
        [c<(areccode+8)⇒[a next← smashcode; next← c. ⇑false]  "assignment"
        user notify: "assigning into read-only field"]]].
    self exstack: 1.
    c=supercode⇒[a next← selfcode; next← supercode] a next← c]'!
Compiler understands: 'encoderef: a | t
    [[trace⇒[user show: a asString]].
    a is: UniqueString⇒
        [t ← cdict lookup: a ⇒ [⇑t]
        a isinfix or: a iskeyword⇒
            [user notify: "missing receiver before: " + a asString]
        cdict insert: a with: (t← self newref: a).  ⇑t]
    ⇑self encodelit: a]'!
Compiler understands: 'newref: a | i
    [for: i from: environment,Undeclared  do:
        [i has: a⇒
            [⇑liticode + (self addlit: (i ref: a))]]
    user show: a asString + " is undeclared.".
    Undeclared define: a  as: nil.
    ⇑liticode + (self addlit: (Undeclared ref: a))]'!
Compiler understands: 'encodelit: a | b i
    [   [a class@Integer⇒[0≠(i← ☞("1 0 1 2 10) find: a)⇒
                [⇑constcode+i-1]
            for: i to: (32 min: literals length) do:
                [literals`i is: Integer⇒[literals`i=a⇒[⇑litcode+i-1]]]];
        @Float⇒[for: i to: (32 min: literals length) do:
                [literals`i is: Float⇒[literals`i=a⇒[⇑litcode+i-1]]]]].
    literals length<32⇒[⇑litcode +(self addlit: a)]
    b ← ObjectReference new. b value← a.
    ⇑liticode + (self addlit: b)]'!
Compiler understands: 'addlit: a | b
    [b ← literals length.
    b≥48⇒[user notify: "too many literals"]
    literals ← literals , a. ⇑b]'!
Compiler understands: 'encodeop: a
    [[trace⇒[user show: a asString]].
    opdict has: a⇒[⇑opdict`a]
    opdict insert: a with: (opcode + (self addlit: a)).
    ⇑opdict`a]'!
Compiler understands: 'remote: b | a t
    [a←Stream default.
    self push. b ← b eval. self pop.     "extra stack to push caller"
    a next← curcode; next← self encodeop: ☞remoteCopy.
    t ← b length+3. a next← t/256+jmpcode+4; next← t\256.
    a append: b; next← endcode; append: (self encodejmp: jmpcode by: 0-t).
    ⇑a contents]'!
Compiler understands: 'push
    [maxstack ← maxstack max: (stack ← stack+1).
    trace⇒[stack print]]'!
Compiler understands: 'pop
    [0>(stack ← stack-1)⇒[user notify: "negative stack"]]'!
```

```
Compiler understands: 'track: expr | a b val
    [a←maxstack.  maxstack← b← stack.
    val ← expr eval.
    b≠stack⇒[user notify: "unbalanced stack"]
    stackused ← maxstack-stack.  maxstack ← a.
    ⇑val]'!
Compiler understands: 'exstack: x
    [maxstack ← maxstack max: stack + x]'!
Compiler understands: 'encodejmp: a by: n | b
    [1≤n and: n≤8⇒[b←String new: 1.
        b˙1 ← n+[a=bfpcode⇒[shortbfp] shortjmp]-1. ⇑b]
    [n<0⇒[n←n+1024. n<0⇒[user notify: "block too long."]]
        a ← a+4. n>1023⇒[user notify: "block too long."]].
    b ← String new: 2.
    b˙1 ← n/256+ a.  b˙2 ← n\256.  ⇑b]'!
Compiler understands: 'compilefor | a b
    [a ← source next.
    [cdict has: a⇒[cdict˙a<litcode⇒[]
            user show: a asString+" is not local."]
    user show: a asString+" is not local."].
    [source ☞to: ⇒
        [code append: (self compileterm: source). self push.
        code next← self encodeop: ☞oneToMeAsStream]
    source ☞from: ⇒
        [code append: (self compileterm: source). self push.
        code next← self encodeop: ☞asStream]
    user notify: "to: or from: expected at "+source peek asString].
    code next← smashcode.
    b← code loc. code next← self newtemp. "reentry point"
    code next← self encodeop: ☞next; next← smashcode.
    code next← self encoderef: a. self compileloop: b.
    ntemps ← ntemps-1]'!
Compiler understands: 'compileuntil: a | b
    [b ← code loc.
    code append: (self compileterm: source). self push.
    [a⇒[code next← self encoderef: ☞false. self push.
        code next← self encodeop: ☞@. self pop]].
    self compileloop: b]'!
Compiler understands: 'compileloop: b|a c
    [self pop.
    [source ☞do: ⇒[]
        user notify: "do: expected before "+source peek asString].
    c ← Compiler new. c compileblock: source.
    c popopt. a ← c code.        "end with pop [opt]"
    code append: (self encodejmp: bfpcode by: 2+a length)."2 for jmp ~n"
    code append: a; append: (self encodejmp: jmpcode by: b-code loc-2)]'!
Compiler understands: 'code [⇑code contents]'!
```

"SCHEDULING"

```
Class new title:'PriorityScheduler';
fields: 'source "the source of power, ie the source from which
        this scheduler was spawned, and who therefore holds
        the suspension if it is suspended."
    processes "<Vector of Contexts> the suspended processes"
    rootprocesses "<Vector of Contexts> root processes for restarting"
    enabled "<Integer> bits for each level; bit0=level0=lowest prio"
    active "<Integer> bits for each level; bit0=1, bit15=sign bit"
    currentlevel "<Integer> this level is currently selected"'~
"The underlying machine has a pointer to the top level scheduler, so that physical
interrupts can also cause interrupts in Smalltalk.  This they do by calling their own
copy of wakeup and reselect.  This copy is also invoked (primitive 65) when reselect is
sent to the top-level scheduler, since its source is the virtual machine itself."
PriorityScheduler understands: 'install: context at: level
    [context sender ← nil.  rootprocesses·level ← context.
    processes·level ← context copy]'!
PriorityScheduler understands: 'enable: level
    [enabled ← enabled lor: biton·level]'!
PriorityScheduler understands: 'disable: level
    [enabled ← enabled land: bitoff·level]'!
PriorityScheduler understands: 'wakeup: level
    [active ← active lor: biton·level.
    self reselect]'!
PriorityScheduler understands: 'sleep: level
    [active ← active land: bitoff·level.
    self reselect]'!
PriorityScheduler understands: 'reset: level
    [processes·level ← rootprocesses·level copy.
    self sleep: level]'!
PriorityScheduler understands: 'restart: level
    [processes·level ← rootprocesses·level copy.
    self wakeup: level]'!
PriorityScheduler understands: 'terminate: level
    [active ← active land: bitoff·level.
    processes·level ← rootprocesses·level ← nil.
    active ← active land: bitoff·level.
    self reselect]'!
PriorityScheduler understands: 'reselect
    [processes·currentlevel ← source current.
    currentlevel ← (active land: enabled) hibit.
    source current ← processes·currentlevel] primitive: 65'!
PriorityScheduler understands: 'current
    [⇑processes·currentlevel]'!
PriorityScheduler understands: 'current ← context
    [⇑processes·currentlevel ← context]'!
PriorityScheduler understands: 'critical: expr| t v
    [t ← self disable.
    v ← expr eval.          "evaluate with no interrupts"
    enabled ← t. ⇑v]'!
PriorityScheduler understands: 'disable | t
    [t ← enabled. enabled ← 0. ⇑t] primitive: 66'!
"remaining here:
    wakeup source on wakeup
    sleep source on sleep
    arrange to sleep or restart when a process returns
    what does an empty scheduler do"
```

"TERMINAL I/O"

```
Class new title: 'UserView';
    fields: 'screenrect "<Rectangle> current screen size"
        vtab "<Integer=0mod2> offset from hardware top"
        htab "<Integer=0mod16> offset from hardware left"
        scale "<Integer=1 or 2> 2 means double bits mode"
        color "<Integer=0 or 1> 1 means reverse field"
        screenfile "<File> for saving current bitmap"
        disp "<Dispframe> default message stream"
        sched "<PriorityScheduler> tasks for this view"'~

"screen overlays"
UserView understands: 'install
    [user close. self reconfigure. user ← self open]'!
UserView understands: 'reconfigure [] primitive: 62'!"tell OOZE and others"
UserView understands: 'close
    [screenfile open.       "write out the old bitmap"
    screenfile write: screenrect height*(screenrect width+15/16)/256
        from:  mem˙066.
    screenfile close]'!
UserView understands: 'open
    [screenfile open.       "read in the new bitmap"
    screenfile read: screenrect height*(screenrect width+15/16)/256
        to:  mem˙066.
    screenfile close]'!

"terminal interaction"
UserView understands: 'kbd [↑kbMap˙self rawkbd]'!
UserView understands: 'kbck
    [self rawkbck⇒[↑kbMap˙self rawkbck] ↑false ]'!
UserView understands: 'rawkbd [] primitive: 63'!
UserView understands: 'rawkbck [] primitive: 64'!
UserView understands: 'buttons
    [↑7-(mem˙0177030 land: 7)]'!
UserView understands: 'redbug [↑self buttons=4]'!
UserView understands: 'yellowbug [↑self buttons=1]'!
UserView understands: 'bluebug [↑self buttons=2]'!
UserView understands: 'anybug [↑self buttons>0]'!
UserView understands: 'keyset
    [↑037-((mem˙0177030 lshift: ¯3) land: 37)]'!
UserView understands: 'mp
    [↑Point new x: mem˙0424-htab y: mem˙0425-vtab]'!
UserView understands: 'cursorloc ← pt
    [mem˙0426 ← pt x+htab. mem˙0427 ← pt y+vtab]'!

"display control"
UserView understands: 'loadcursorbitsfrom: bitstr | i
    [for⁸ i to: 16 do⁸
        [mem˙(0430+i) ← bitstr word: i]]'!
UserView understands: 'displayoffwhile⁸ expr | t v
    [t ← mem˙067. mem˙067 ← 60.
    v ← expr eval. mem˙067 ← t. ↑v]'!
UserView understands: 'restoredisplay
    [mem˙067 ← screenrect height/2]'!
UserView understands: 'screenextent: extent tab: tab
    [mem˙065 ← (0400*(tab x/16))+(extent x/16|2).
    mem˙067 ← extent y/2. mem˙063 ← tab y/2.
    htab ← tab x|16. vtab ← tab y|2.
    screenrect ← 0⊚0 rect: (extent x|32)⊚(extent y|2).
    self reconfigure]'!

"system messages"
UserView understands: 'ev [disp ev]'!
```

```
UserView understands: 'request: s [⇑disp request: s]'!
UserView understands: 'show: str
    [disp append: str; show]'!
UserView understands: 'cr [disp cr]'!
UserView understands: 'clearshow: str
    [disp clear; append: str; show]'!
UserView understands: 'notify: str
    [self restoredisplay. disp append: str; cr; show.
    disp append: thisContext sender trace; show.
    thisContext sender debug. self restart]'!
UserView understands: 'restart
    [self restart§ [disp ev]]'!
UserView understands: 'restart§ code
    [code sender ← nil.
    thisContext sender ← nil.        "release caller chain"
    CodeKeeper ← (Vector new: 10) asStream.    "release held code"
    disp cr; append: " ** restart"; show.
    while§ true do§ [code eval]]'!
UserView understands: 'screenrect [⇑screenrect]'!

"window scheduling"
UserView understands: 'schedule: window
    [sched@nil⇒
        [(sched ← Vector new: 1)·1 ← window]
    sched ← sched , window]'!
UserView understands: 'unschedule: window
    [0<(t← sched find: window)⇒
        [sched ← sched·(1 to: t-1) concat: sched·(t+1 to: sched length)]]'!
UserView understands: 'sched [⇑sched]'!
UserView understands: 'run | s
    [while§ (screenrect has: user mp) do§
        [for§ s from: sched do§ [s startup]]]'!
UserView understands: 'printon: strm
    [strm append: "a UserView"]'!

"OOZE save/load"
UserView understands: 'OutLd [] primitive: 85'!
    "returns true from call; false from OS resume"
UserView understands: 'InLd: fileid
    [user notify: "file problem"] primitive: 86'!
UserView understands: 'overlay: fileid
    [self OutLd⇒[self InLd: fileid]]'!
UserView understands: 'quit | t
    [t ← Vector new: 5; all ← 0.  self overlay: t]'!
UserView understands: 'Swat [] primitive: 90'!
```

"PARAGRAPH EDITOR"

```
Smalltalk define ☞Peditorstuff as ☞Peditorstuff ← SymbolTable new.!
Peditorstuff insertall: ☞(bs del esc doit paste typing editmenu
      Scrap Deletion ctlchars runvals p1 p2 selon oldpara oldheight)
   with: ☞(010 0177 033 036 02 typing editmenu
      Scrap Deletion ctlchars runvals p1 p2 selon oldpara oldheight).!
Class new title: 'ParagraphEditor';
   subclassof: Textframe;
   fields: 'loc1 loc2';
   sharing: 'Peditorstuff'
ParagraphEditor understands: 'enter: window
   [frame ← window inset: 4 ☺ 2 and: 2 ☺ 2.
   oldpara ← oldheight ← false. self show]'!
ParagraphEditor understands: 'leave
   [self comp: p1 to: p2]'!

ParagraphEditor understands: 'kbd: boss | more char
   [more ← Stream default.
   while☺ user kbck do☺
      [(char ← user kbd)=del⇒ []
      char=bs⇒ [more empty⇒[loc1 ← 1 max: loc1-1]."backspace"
            more skip: ¯1]
      more next← char].
   self replace: more contents]'!

ParagraphEditor understands: 'redbug: boss | a t drag2
   [t ← self charnearpt: user mp.
   t=loc1 and: loc1=loc2 ⇒ [self selectword] " double-bug "
      [loc1=t⇒[drag2 ← false]
      drag2 ← true. loc2=t⇒[]     " pick up old selection "
      self comp: p1 to: p2. loc1 ← loc2 ← t.   " start new selection "
      p2 ← 1☺0 + (p1 ← reply1).
      self comp: p1 to: p2].
   while☺ user redbug do☺                " draw out selection "
      [a ← reply1.
         [loc1=loc2⇒[drag2 ← t≥loc2]].
         [drag2⇒ [ [t<loc1⇒[t←loc1. a←self ptofchar: t]].
               self comp: a to: p2. loc2 ← t. p2 ← a]
            [t>loc2⇒[t←loc2. a←self ptofchar: t]].
            self comp: p1 to: a. loc1 ← t. p1 ← a].
         [p1=p2⇒[self comp: p1 to: (p2 ← 1☺0 + p1)]].
      t ← self charnearpt: user mp]]'!

ParagraphEditor understands: 'replace: t
   [  [oldpara⇒[] oldpara ← para].
   para ← para replace: loc1 to: loc2-1 by: t.
   loc1 ← loc2 ← loc1 + t length.
   self show]'!
ParagraphEditor understands: 'selection
   [↑para copy: loc1 to: loc2-1]'!

ParagraphEditor understands: 'show
   [super show.
   [loc1@nil⇒ [loc1←loc2←1]].
   user kbck⇒[] self select]'!
ParagraphEditor understands: 'select
   [p1 ← self ptofchar: loc1.
   p2 ← [loc2=loc1⇒[1☺0 + p1] self ptofchar: loc2].
   self comp: p1 to: p2]'!

ParagraphEditor understands: 'comp: a to: b | t"complement from a to b"
   [  [a y<b y⇒[] (a y=b y) and: (a x<b x)⇒[] t←a. a←b. b←t].    "in case they were reversed"
      [a y < b y⇒
         [(a rect: (window corner x-4) ☺ (a y+self lineheight)) comp."top line"
         a ← (window origin x+4) ☺ (a y+self lineheight).
         a y < b y⇒
            [(a rect: (window corner x-4) ☺ b y) comp]"middle (if any)"
```

```
        ]].
      a y > b y⇒[]
      (a x ⊚ b y rect: b x ⊚ (b y + self lineheight)) comp."bottom (or only)"]'!

ParagraphEditor understands: 'selectword | a b dir t level open close
   [self comp: p1 to: p2.
   a← b← dir← ¯1.
   open ← "([{< """"
". close ← ")]}> """"
".     [loc1>para length⇒[]
          loc1≤1⇒[dir←1]
          t←open find: (a←para˙(loc1-1)). t>0⇒  "delim on left"
             [dir←1. b←closet.      "match to the right"
             a=b⇒[a←¯2]]        "non-nesting"
          t←close find: (a←para˙loc1). t>0⇒   "delim on right"
             [dir←¯1. b←opent.     "match to the left"
             a=b⇒[a←¯2]]        "non-nesting"
          a← ¯1].       "no delims - select a token"
       t←loc1-1. level←1.
       until⁸ [t≤1⇒[] t≥para length⇒[] level=0⇒[] false] do⁸
          [para˙(t← t+dir) = b⇒ [level← level-1];    "leaving nest"
          = a⇒ [level← level+1].   "entering nest"
          a=¯1⇒[(parat) tokenish⇒[]    "token check goes left "
                dir=¯1⇒[loc1 ← t+1. dir←1. t←loc2]"then right"
                level← 0]]
   [dir=1⇒[loc2← t] loc1← t+1].
   self select]'!
Integer understands: 'tokenish   "test for token-chars" .
   [self isletter⇒[⇑true]"lower-case"
    self isdigit⇒[⇑true]"digits"
    ⇑"¯.:⁸" has: self]'!  "also high-minus and dot"
```

"CLASS EDITOR"

```
Class new title: 'ClassEditor';
   subclassof: ParagraphEditor;
   fields: 'class selector'
ClassEditor understands: 'firsttime
   [window has: user mp⇒
      [window outline. self enter: window]
   ⇑false]'!
ClassEditor understands: 'eachtime
   [window has: user mp⇒
      [user kbck⇒[self kbd: nil]
      user anybug⇒
         [user redbug⇒[⇑self redbug: nil]
         user yellowbug⇒[self yellowbug: nil]
         user bluebug⇒[window ← Rectangle new fromuser.
               self firsttime]]]
   user yellowbug⇒[self close. ⇑false]
   user anybug⇒[self leave. ⇑false]]'!
ClassEditor understands: 'lasttime
   [self leave]'!

ClassEditor understands: 'class: class selector: selector
      para: para window: window
   [para ← para asParagraph. style ← DefaultTextStyle.
   self enter: window]'!
ClassEditor understands: 'yellowbug: boss
   [class understands: para]'!
ClassEditor understands: 'close
   [(window inset: ¯2 @ ¯2) clear: backround.
   user unschedule: self]'!

Class understands: 'edit: selector
   [user schedule: (ClassEditor new class: self selector: selector
         para: (self code: selector) window: Rectangle new fromuser)]'!

Dispframe understands: 'firsttime
   [text window has: user mp⇒
      [text window outline. self cr; append: "🕭"; show]
   ⇑false]'!
Dispframe understands: 'eachtime
   [text window has: user mp⇒
      [user kbck⇒[t← self read.
         self print: [t@nil⇒[self doit] nil'st].
         self cr; append: "🕭"; show]]
   user anybug⇒[⇑false]]'!
Dispframe understands: 'lasttime
   [self skip: [¯2 max: 0-position]; show]'!
```

****** NOT DONE BEYOND HERE ******

```
"OTHER PARAGRAPH STUFF"
ParagraphEditor understands: 'kbdlook | t c
   [(c← user rawkbck)>170⇒
      [t← ctlchars find: c.
      t>0⇒[kbd. changed ← true.
            para changestyle: runvals˙t from: loc1+1 to: loc2]
      ⇑false]
   ⇑false]'!
Peditorstuff insertall: ☞(ctlchars runvals) with: ☞(
      (226 233 224 173 242  "ctl-  b i ... - r"
         176 177 178 179 180 181 182 183 184 185"ctl-  0 thru 9"
         193 194 195 196 197 198)              "ctl-  A thru F"
      (1 2 4 8 ¯1
         0 1 2 3 4 5 6 7 8 9
         10 11 12 13 14 15)
      ).!
☞t ← Peditorstuff˙☞runvals.
   for i ← 6 to t length do
      (t[a] ← 16*t[a]).!    "font number left 4 bits"

ParagraphEditor understands: 'readchange
   [self cleanup. changed←false. ⇑para]'!
ParagraphEditor understands: 'fixframe: x
   [⇑x]'!
ParagraphEditor understands: 'close: x []'!
ParagraphEditor understands: 'outside: x [⇑false]'!
ParagraphEditor understands: 'asParagraph[⇑para]'!
ParagraphEditor understands: 'changed[⇑changed]'!
ParagraphEditor understands: 'selection[⇑para copy: loc1 to: loc2-1]'!
```

"WINDOW FRAME AND CONTROL"

```
Class new title: 'UserWindow';
   fields: 'frame contents title';
   sharing: Userstuff
UserWindow understands: 'frame: f contents: contents
   [frame ← contents fixframe f. self show]'!
UserWindow understands: 'of: c
   [self frame: Rectangle new fromuser contents: c]'!
UserWindow understands: 'show
   [self showin: user screenrect]'!
UserWindow understands: 'showin: r | a
      [(a ← frame intersect: r) empty⇒ []
      a color: dkgray mode: storing.
      ((frame inset: p1 and: p2) intersect: r) color: white mode: storing.
      contents enter: r.
      whitey show: contents title clipped: r]

UserWindow understands: 'firsttime
   [frame has: user mp⇒ [self show] ⇑false]'!
UserWindow understands: 'eachtime
   [frame has: user mp⇒
      [user buttonck⇒
         [user redbug⇒[⇑contents redbug: self]
         user yellowbug⇒[⇑contents yellowbug: self]
         user bluebug⇒[⇑contents bluebug: self]]]
      user kbck⇒[contents kbd: self]
      user keysetck⇒[contents keyset: self]]
   contents outside: self⇒[]
   user buttonck⇒[frame has mp⇒[] ⇑false]
   user kbck⇒[user kbd; user flash] "flush typing outside"]'!
UserWindow understands: 'lasttime
   [⇑contents leave]'!

UserWindow understands: 'newframe
   [while§ user buttonck do§ [].
   frame ← Rectangle new origin: (a← user mp) corner: a.
   while§ user buttonck do§
      [frame color: dkgray mode: xoring.
      frame ← contents fixframe: (frame growto: user mp).
      frame color: dkgray mode: xoring].
   frame color: dkgray mode: storing.
   whitey show: contents title.
   windowSched fillholes: (oldframe nonintersect: frame).
   ⇑frame]'!
UserWindow understands: 'close
   [contents close: self]'!
UserWindow understands: 'movegrow
   [frame color: white mode: storing.
   frame ← self newframe.
   ☞a ← frame inset p1 p2.
   a paint 12 white.
   contents enter a inset p3 p4]'!
"***"
UserWindow evals [☞p1 ← point 3 fontheight+4. ☞p2 ← point 3 3.
      ☞p3 ← point 4 2. ☞p4 ← point 4 0. ☞p5 ← point 3 2.
      ☞whitey ← textframe rectangle p1 p1]'!
```

"FORMATTED TEXT"

```
Paragraph understands: 'runfind b | a      "**index into run**"
   [a←1.
   while☺ true do☺
      [runs'a≥b⇒ [⇑a , b]      "return run, index"
      b ← b-(runs'a). a ← a+2]]'!
Paragraph understands: 'runs      "**subrange of run**"
   [◀[⇒[☞a ← :.      "indexed by text, not run"
         ◀to⇒
            [☞b ← [◀end⇒[☞c←runs length-1. point c runs[c]]
                  self runfind :]. ◀].
            a>text length⇒[⇑""] ☞a ← self runfind a.
            ☞c ← runs[a x to b x+1].      "copy the sub-run"
            c length=0⇒[⇑c]
            a x = b x⇒[0= c[1]← 1+ b y-a y ⇒[⇑""] ⇑c]
            c[1] ← 1+runs[a x] - a y.          "trim the end lengths"
            c[c length-1] ← b y.  ⇑c]
         ◀]. ⇑runs[self runfind a x+1]]
      ⇑runs]'!
string understands ☞runcat "**concatenate runs**"
   [☞x←:. x length=0⇒[] self length=0⇒[⇑x]
      self[self length]=x[2]⇒
         [⇑self[1 to self length-2]+
            [makerun self[self length-1]+x[1] x[2]]+
            x[3 to x length]]
      ⇑self + x]'!
Paragraph understands: 'asString [⇑text]'!
Paragraph understands: '[      "**subranges and subscripting of text**"
   [☞a←:.
      ◀to⇒[☞b←:. ◀].
         ◀replace⇒[☞c←:."**alters self; doesn't copy"
            ☞runs ← self runs[1 to a-1]
               runcat [c is string⇒[makerun c length self runs[b]] c runs]
               runcat self runs[b+1 to end].
            ☞text ← text[a to b] replace
               [c is string⇒[c] c text]]
         ◀changestyle⇒[☞c←:.
            ☞runs ← self runs[1 to a-1]
               runcat [self mergestyle c into self runs[a to b]]
               runcat self runs[b+1 to end]]
         ◀←⇒[◀all⇒[text[a to b] ← all :]
               text[a to b] ← :]
         ⇑paragraph with text[a to b] self runs[a to b]]"**copy align**"
      ◀]. ◀←⇒[⇑text[a]←:]
      ⇑text[a]]'!
Paragraph understands: 'mergestyle
   [☞a←:. ◀into. ☞b←:.
      for c← 2 to b length by 2 do
         [b[c] ← [a=1⇒ [0]          "reset"
            0<a≤017⇒ [a ⊟b[c]]      "toggle emphasis"
            a+017 ⊞b[c]]].   "new font"
      ⇑b]!

☞makerun ← function [run len str] [☞len←:. ☞run←:.
      len=0⇒[⇑""] ☞str←stream. repeat do
         [run<256⇒[str←len. str←run. ⇑str contents]
         str←255. str←run. ☞len←len-255]]'!
paragraph'sinitcode ←
   [☞alignment ← 0. ◀with⇒[☞text ← :.
         ☞runs ← [◀onerun⇒[makerun text length :] :]]
      ☞text ← string 0]'!
Paragraph understands: 'runshow | strm i"**print a run**"
   [strm ← stream. for☺ i from: (1 to: runs length by: 2) do☺
      [strm append: (runs'i) asString; append: '.'.
      strm append: (runs'(i+1)) asString; append: ', ']
   ⇑strm contents]'!
```

"MENUS"

```
☞menu ← class [[pt i bits][str text thisline frame but][pt2][]] [] [] []'!

menu'sinitcode ←
   [☞str←:. ☞but←:.
   ☞text ← textframe rectangle ☞pt←point 0 0 point 1000 1000 with str.
   for i to str length+1 do
      [☞pt ← pt max text ptofchar i]
   text'sframe growto pt+point 4 text fontheight.
   text'spara center.
   ☞frame ← text'sframe inset ☞pt2←point ~2 ~2 pt2.
   ☞thisline←rectangle text's frame'sorigin
            point text'sframe'sextent x text fontheight]'!
menu understands: 'bug
   [☞pt←mp-thisline center.    "center prev item on mouse"
      text'sframe moveby pt. thisline moveby pt.
      frame moveby pt. ☞bits ← frame makebuff."save background"
      frame paint 12 ~1. text show.
      0=mouse 7⇒[frame loadbuff bits. ⇑0]     "accidental bug returns 0"
      thisline comp.
      repeat do [
         frame has ☞pt←mp⇒    "in the menu"
            [button but⇒          "button still down"
               [thisline has pt⇒[]
               text charnearpt pt. ☞pt←text reply.
               thisline comp. thisline moveto"selection follows mouse"
                    point text'sframe'sorigin x pt y.
               thisline comp]
            frame loadbuff bits. "restore background, return index"
            ⇑1+[thisline'sorigin-frame'sorigin]y/text fontheight]
            thisline comp.     "he left the menu"
         repeat do [button but⇒[frame has mp⇒[thisline comp. done]]
                  frame loadbuff bits. ⇑0]"return 0 for abort"
      ]]'!
menu understands: 'rebug
   [showcursor [but=1⇒[bug2cursor] bug3cursor]
      repeat do [button but⇒[done]]  "wait for button down again"
      ⇑self bug]'!
```

"BRAVO CONVERSION"

```
☞bravo2para ← function [a b c f ff i p q s st]
    [☞f←:. f end⇒[⇑false]
    ☞a ← f upto 26. f next.    "pick up ascii"
    ☞p ← paragraph.
    ☞b ← f upto 13. f next.       "pick up trailer"
    ☞s ← stream of b. ☞st ← stream.
    repeat do
       [☞c ← s next.              "scan for useful para info"
       c='c'[1]⇒[p center]
       c='j'[1]⇒[p justify]
       c='\'[1]⇒[done]
       st ← c]                 "stache rest for later bravo output"
    "p'strailer ← st contents." st reset.
    ☞ff← ☞len ← ☞run ← 0. ☞q←1.
    repeat do
       [s end⇒[done]
       ☞c ← s next.
       0< ☞i← 'bBiIuUnNfoy'[1 to 11] find c ⇒
       [☞[[☞run ← run+1]      [☞run ← run-1]     "bold"
          [☞run ← run+2]     [☞run ← run-2]     "italic"
          [☞run ← run+4]     [☞run ← run-4]     "underline"
          [☞run ← run+8]     [☞run ← run-8]     "strikeout"
          [☞run ← [run ⍟017] + 16*☞ff← scannum s]"font"
          [☞r←scannum s. ☞run ← [run ⍟017] +      "super/sub script"
             16*[r=0⇒[ff] r>128⇒[11] 10]]          "later vary by font"
          [scannum s]]     "tab color"
          [i] eval]
       060≤c≤071⇒[s skip ¯1. ☞r ← scannum s."get run length"
          ☞len ← len+r.     "later merge across ignored changes"
          st ← makerun r run]]
    [0< ☞r← a length-len⇒
       [st ← makerun r run]]
    p'schars ← a. p'sruns ← st contents.  ⇑p]'!
☞scannum ← function [f c i] [☞f←:. ☞i←0.
  repeat do
    [☞c←f next.
    060≤c≤071⇒[☞i← [10*i] + c-060]
    f skip ¯1. done]
  ⇑i]'!
```

```
"FILES AND DIRECTORIES
June 15, 1977  1:18 PM"

Integer understands: 'allmask: b [⇑b = (self land: b)]'!
Integer understands: 'anymask: b [⇑0 ≠ (self land: b)]'!
Integer understands: 'nomask: b [⇑0 = (self land: b)]'!
Integer understands: 'asLowercase
   [0101 ≤ self⇒ [
      self ≤ 0132⇒ [⇑self + 040]]]'!
Integer understands: 'asUppercase
   [0141 ≤ self⇒ [
      self ≤ 0172⇒ [⇑self - 040]]]'!
Integer understands: 'compareChar: c
   [⇑self asLowercase compare: c asLowercase]'!

Number understands: 'compare: i
   [self < i⇒ [⇑1]
   self = i⇒ [⇑2]
   ⇑3]'!

String understands: 'compare: s | i v
   [for: i to: (self length min: s length) do:
      [self˙i = (s˙i)⇒ []
      v ← self˙i compareChar: s˙i.
      v ≠ 2⇒ [⇑v]]
   ⇑self length compare: s length]'!

String understands: '< s [⇑1 = (self compare: s)]'!
String understands: '= s [
   self length = s length⇒ [⇑2 = (self compare: s)]
   ⇑false]'!
String understands: '> s [⇑3 = (self compare: s)]'!

UserView understands: 'read [⇑disp read]'!


Smalltalk define ☞Filestuff as ☞Filestuff ← SymbolTable new.!
"disk commands"
Filestuff insertall: ☞(CRR CRW CCR CCW CWW)
   with: ☞(044100 044110 044120 044130 044150)!
"disk label"
Filestuff insertall: ☞(nextp backp lnused numch pagen version sn1 sn2)
   with: ☞(1 2 3 4 5 6 7 8)!
Filestuff insertall: ☞(read write shorten) with: ☞(1 2 4)!
Filestuff insertall: ☞(old oldornew new created) with: ☞(1 2 3 4)!
Filestuff insertall: ☞(dfmask boffset) with: ☞(02000 040)!
Filestuff insertall: ☞(dirname defdir pagelength) with: ☞('SysDir.' nil 512)!


Class new title: 'File';
   subclassof: Stream; "except end, pastend (←), reset, contents, skip"
   fields: "array position limit" 'dirinst filename rvec label rwmode leader curadr status';
   sharing: 'Filestuff'!

File understands: 'readonly [rwmode ← read]'!
File understands: 'readwrite [rwmode ← read + write]'!
File understands: 'writeonly [rwmode ← write]'!

File understands: 'old [status ← old]'!
File understands: 'oldornew [status ← oldornew]'!
File understands: 'new [status ← new]'!

File understands: 'on: dirinst []'!
File understands: 'random [rvec ← Vector new: 0]'!

File understands: 'dskio: curadr
      command: command
      page: startpage
      pages: npages
      onerror: exp
   [⇑exp eval] primitive: 80'!
```

```
File understands: 'docommand: com [
    self dskio: curadr
        command: com
        page: label'pagen
        pages: 1
        onerror: [↑self error: "docommand"]]'!

File understands: 'error: s [
    user notify: "File error: "+s+" in "+filename asString.
    ↑false]'!

File understands: 'named: filename [
    filename ← self namecheck: filename⇒ [↑self find]
    ↑false]'!

File understands: 'namecheck: name | i x [
    name length = 0⇒ [↑self error: "empty name"]
    name length > 255⇒ [↑self error: "name too long"]
    for: i to: name length do: [
        "check characters"
        x ← name'i.
        x isletter⇒ []
        x isdigit⇒ []
        0 = ("+-()↑⇒." find: x)⇒ [
            ↑self error: "illegal character " + (name'(i to: i))]]
    x ≠ ("."'1)⇒ [↑name + "."]
    ↑name]'!

File understands: 'getname: name | t [
    user show: "type a file name".
    user show: [name length > 0⇒ [", just ! for " + name] " "]
    t ← user read.
    t @ false⇒ [↑false]
    ↑self named: [t @ nil⇒ [name] t]]'!

File understands: 'reopen [
    self dskio: leader
        command: CCR
        page: 0
        pages: 1
        onerror: [
            "leader page failed"
            self oldornew.
            ↑self find].
    filename ≠ (array'(14 to: array'13 + 13))⇒ [
        "name failed - how about ser no?"
        self oldornew.
        ↑self find]

    [rvec @ nil⇒ []
    self random].
    self old.
    self init]'!

File understands: 'find | entry [
    "some defaults"
    [status @ nil⇒ [self oldornew]].
    [rwmode @ nil⇒ [self readwrite]].
    [dirinst @ nil⇒ [self on: defdir]].
    [array @ nil⇒ [self of: (String new: pagelength)]].
    label ← Vector new: 8.
    label'nextp ← 0.
    label'backp ← 0.
    label'lnused ← 0.
    label'numch ← pagelength.
    label'pagen ← 0.
    entry ← dirinst find: filename⇒ [
        status = new⇒ [↑false]
        label'sn1 ← entry word: 2.
        label'sn2 ← entry word: 3.
```

```
        label'version ← entry word: 4.
        self dskio: (leader ← self virtualToAlto: (entry word: 6))
            command: CCR
            page: 0
            pages: 1
            onerror: [↑self error: "leader page"].
        self init]
    status = old⇒ [↑false]
    rwmode nomask: write⇒ [↑false]
    (leader ← dirinst alloc: 0) @ false⇒ [↑false]
    entry ← dirinst getNewSn.
    label'sn1 ← entry word: 1.
    label'sn2 ← entry word: 2.
    label'version ← 1.

    status ← created "for leaderstamp".
    "6 header words, length char, filename, possible null"
    entry ← String new: 13 + (filename length lor: 1).
    entry word: 1 ← entry length/2 lor: dfmask.
    entry word: 2 ← label'sn1.
    entry word: 3 ← label'sn2.
    entry word: 4 ← label'version.
    entry word: 5 ← 0 "?".
    entry word: 6 ← self altoToVirtual: leader.
    entry'13 ← filename length.
    entry'(14 to: 13+filename length) ← filename.
    self init.
    dirinst next ← entry]'!

File understands: 'init | oldmode s [
    s ← String new: 4.
    s word: 1 ← mem'0572.
    s word: 2 ← mem'0573.
    oldmode ← rwmode.
    self readwrite.

    "assumes positioned at page 0"
    position ← 0.
    [status ≠ created⇒ [self skip: 4]
    self append: s]."creation date"

    [oldmode allmask: write⇒ [self append: s]"write date"
    self skip: 4].

    "read date"
    self append: s.

    [status = created⇒ [
        self next ← filename length."file name (Bcpl style)"
        self append: filename]].

    [rvec @ nil⇒ []
    "random access"
    self settoend.
    [label'pagen > rvec length⇒ [
        rvec ← rvec copyto: (Vector new: label'pagen)]].
    for: s to: rvec length do: [
        rvec's @ nil⇒ [
            self settopage: s char: 0.
            rvec's ← curadr]]].

    self reset.
    rwmode ← oldmode]'!

File understands: 'flush [
    rwmode nomask: write⇒ [↑false]
    label'numch < position⇒ [
        position ≠ pagelength⇒ [self docommand: CWW]
        label'nextp = 0⇒ [self extendto: label'pagen + 1]
        self docommand: CWW]
    self docommand: CCW]'!
```

```
File understands: 'settopage: page char: char | pch pn [
   pch ← [char<0⇒ [0-char] char].
   page ← pch/pagelength + page.
   pch ← pch \ pagelength.
   [char < 0 and: pch ≠ 0⇒[
      pch ← pagelength - pch.
      page ← page-1]].
   page < 0⇒ [⇑self error: "negative page"]
   "possibly write current page"
   [label'pagen < page and° label'nextp = 0⇒ [self extendto: page]
   self flush].
   "try random access"
   [rvec @ nil⇒ []
   label'pagen = page⇒ ["already there"]
   "chain read from the nearest page we can"
   pn ← page min: rvec length.
   while° [pn > 0 and° rvec'pn @ nil] do° [pn ← pn - 1]
   self dskio: [pn = 0⇒ [leader] rvec'pn]
      command: CCR page: pn pages: page+1-pn
      onerror° [⇑self error: "random access"]].
   "do sequential reads to get there, possibly extending file"
   [label'pagen < page⇒ [
      "chained read forward"
      [label'nextp ≠ 0⇒ [
         self dskio: label'nextp
            command: CCR
            page: label'pagen+1
            pages: page-(label'pagen)
            onerror° [⇑self error: "forward chaining"]]].
      label'pagen = page⇒ ["finished"]
      rwmode allmask: write⇒ [self extendto: page]
      "change the request"
      page ← label'pagen]
   label'pagen > page⇒
      [label'pagen-1 = page⇒
         ["one page backwards"
         self dskio: label'backp
            command: CCR page: page pages: 1
            onerror° [⇑self error: "reading backwards"]]
      "forward from leader"
      self dskio: leader
         command: CCR
         page: 0
         pages: page+1
         onerror° [⇑self error: "leader reading"]]].

   label'pagen ≠ page⇒ [⇑self error: "didnt get to page"]
   position ← [rwmode allmask: write⇒ [pch] label'numch min: pch].
   limit ← label'numch]'!

File understands: 'end [
   position < (label'numch)⇒ [⇑false]
   ⇑label'nextp = 0]'!

File understands: 'pastend [
   self end⇒ [⇑false]
   self settopage: label'pagen+1 char: 0.
   ⇑self next]'!

File understands: 'pastend ← x [
   [limit < pagelength⇒ [limit ← pagelength]
   self settopage: label'pagen+1 char: 0].
   ⇑self next ← x]'!

File understands: 'reset [self settopage: 1 char: 0]'!

File understands: 'settoend | oldmode [
   self flush.
   oldmode ← rwmode.
   self readonly.
```

pch ← label'numch

```
    self settopage: 5000 char: 0.
    rwmode ← oldmode]'!

File understands: 'contents | s [
    self settoend.
    s ← String new: (label·pagen-1)*pagelength + (label·numch).
    self reset.
    ⇑self into: s]'!

File understands: 'skip: n [
    position ← position + n.
    position < 0 or: position ≥ limit⇒ [
        self settopage: label·pagen char: position]]'!

File understands: 'positioneven [
    position allmask: 1⇒ [self skip: 1]]'!

File understands: 'nextword | s [
    position ← position + 1 | 2.
    self end⇒ [⇑false]
    s ← self into: (String new: 2).
    ⇑s word: 1]'!

File understands: 'nextword ← w | s [
    s ← String new: 2.
    s word: 1 ← w.
    self positioneven.
    self append: s]'!

File understands: 'close [
    rwmode allmask: shorten⇒ [self shortento: label·pagen char: position]
    self flush]'!

File understands: 'extendto: dest [
    rwmode nomask: write⇒ [⇑false]
    [rvec @ nil⇒ []
    rvec ← rvec copyto: (Vector new: dest)].
    until: label·pagen = dest do: [
        label·nextp ← dirinst alloc: curadr.
        label·numch ← pagelength.
        self docommand: CWW.
        label·backp ← curadr.
        curadr ← label·nextp.
        label·pagen ← label·pagen + 1.
        rvec @ nil⇒ []
        rvec·(label·pagen) ← curadr]
    label·nextp ← 0.
    label·numch ← 0.
    position ← 0.
    self docommand: CWW]'!

File understands: 'shortento: page char: char [
    rwmode nomask: write⇒ [⇑false]
    self settopage: page char: char.
    dirinst dealloc: label·nextp.
    label·nextp ← 0.
    label·numch ← label·numch min: char.
    self docommand: CWW.
    rvec @ nil⇒ []
    rvec ← rvec copyto: (Vector new: label·pagen)]'!

File understands: 'delete [
    rwmode allmask: write⇒ [
        dirinst delete: filename⇒ [dirinst dealloc: leader]
        ⇑false]
    ⇑false]'!

File understands: 'rename: newname [
    rwmode nomask: write⇒ [⇑false]
    (newname ← self namecheck: newname) @ false⇒
        [⇑self error: "badnew name"]
```

```
    dirinst find: newname⇒ [⇑self error: newname + " already exists"]
    (dirinst rename: filename to: newname) @ false⇒
        [⇑self error: "file not there"]
    filename ← newname.
    self settopage: 0 char: 12.
    self next ← filename length.
    self append: filename]'!

File understands: 'virtualToAlto: vadr [
    ⇑(vadr \ 12 lshift: 12) + (vadr/12 * 4)]'!

File understands: 'altoToVirtual: dadr [
    ⇑(dadr lshift: ¯12) + (3 * (dadr land: 07774))]'!

File understands: 'altoToKeys: dadr | i k [
    k ← Stream default.
    for§ i to: 14 do§ [
        dadr allmask: (0100000 lshift: 1-i)⇒ [
            k next←"546E7DUV0K-P/\"'i]]
    ⇑k contents]'!

File understands: 'last ← x [
    self skip: ¯1.
    self next ← x]'!

File understands: 'appendlast: x [
    self skip: 0-x length.
    self append: x]'!

File understands: 'zaplabel [
    label'nextp ← 0.
    label'pagen ← 0.
    label'sn1 ← ¯1.
    label'sn2 ← ¯1.
    label'version ← ¯1]'!

File understands: 'fileid | v [
    v ← Vector new: 5.
    v'1 ← label'sn1.
    v'2 ← label'sn2.
    v'3 ← label'version.
    v'4 ← 0.
    v'5 ← curadr.
    ⇑v]'!

File understands: 'curadr [⇑curadr]'!
File understands: 'nextp [⇑nextp]'!

"other unimplemented File messages:
load, save, copyto, length, remove, exists..."

Class new title: 'Directory';
    fields: 'dirinst bitinst filinst junkfile closed';
    sharing: 'Filestuff'!

Directory understands: 'reset [
    "only system directories implemented now"
    closed⇒ [
        closed ← false.
        filinst @ nil⇒ [
            filinst ← [File new readwrite old on: self; named: dirname].
            bitinst ← [File new readwrite old on: self; named: "DiskDescriptor."].
            junkfile ← [File new readwrite old on: self; named: "Com.cm."]]
        filinst reopen.
        bitinst reopen.
        junkfile reopen]
    filinst reset.
    bitinst @ nil ⇒[] bitinst flush]'!

Directory understands: 'open [self reset]'!
```

```
Directory understands: 'File [↑File new on: self]'!

Directory understands: 'close [
    filinst close.
    bitinst close.
    closed ← true]'!

Directory understands: 'asStream'!
Directory understands: 'next | w [
    w ← filinst nextword⇒ [
        filinst skip: ¯2.
        ↑filinst into: (String new: 2 * (w land: dfmask-1))]
    ↑false]'!
Directory understands: 'next ← entry [
    filinst append: entry.
    filinst flush]'!

Directory understands: 'find: name | entry [
    name = dirname⇒ [
        entry ← String new: 12.
        entry word: 2 ← 0100000.
        entry word: 3 ← 0.
        entry word: 4 ← 1.
        entry word: 6 ← 1.
        ↑entry]
    self reset.
    for: entry from: self do: [
        (entry word: 1) allmask: dfmask⇒ [
            "normal entry, check name"
            entry˙13 ≠ name length⇒ []
            name = (entry˙(14 to: entry˙13 + 13))⇒ [↑entry]]
        "deleted entry"]
    ↑false]'!

Directory understands: 'list | entry [
    self reset.
    for: entry from: self do: [
        (entry word: 1) allmask: dfmask⇒ [
            user show: entry˙(14 to: entry˙13 + 13)]
        "ignore deleted entries"]]'!

Directory understands: 'delete: name | entry [
    entry ← self find: name⇒ [
        "mark entry as deleted"
        filinst skip: 0- entry length.
        filinst nextword ← (entry word: 1) land: dfmask-1.
        "filinst skip: ¯2"
        filinst flush]
    ↑false]'!

Directory understands: 'rename: name to: newname | entry nentry [
    entry ← self find: name⇒ [
        "assumes newname already checked to be not there"
        nentry ← String new: (newname length lor: 1) + 13.
        nentry word˙ 1 ← nentry length/2 lor: dfmask.
        nentry˙(3 to: 12) ← entry˙(3 to: 12).
        nentry˙13 ← newname length.
        nentry˙(13 to: 13+newname length) ← newname.
        "mark old entry deleted"
        filinst skip: 0- entry length.
        filinst nextword ← (entry word: 1) land: dfmask-1
        filinst settoend.
        self append: nentry]
    ↑false]'!

Directory understands: 'alloc: dadr | index start stop ch i m [
    "starting place in bittable"
    start ← (bitinst altoToVirtual: dadr) land: 0177770.
    stop ← 4872 "see free".
    for: i to: 2 do: [
        "up to two passes through table"
```

```
        bitinst settopage: 1 char: start/8 + boffset.
        for° index from: (start to: stop by: 8) do° [
            ch ← bitinst next.
            ch = 0377⇒ []
            m ← 0200.
            while° m > 0 do° [
                (ch land: m) ⇒[
                    ch ← ch lor: m.
                    "check if page is really free"
                    junkfile zaplabel.
                    junkfile dskio: (junkfile virtualToAlto: index)
                        command: CRR
                        page: 0
                        pages: 1
                        onerror° [false] ⇒[m←¯1]]
                index ← index + 1.
                m ← m lshift: ¯1]
            "update bittable"
            bitinst last ← ch.
            m = ¯1⇒ [↑junkfile curadr]]
        stop ← start.
        start ← 0]
    bitinst error: ''nofreepages'']'!

Directory understands: 'dealloc: dadr | index ch m [
    until° dadr = 0 do° [
        index ← bitinst altoToVirtual: dadr.
        bitinst settopage: 1 char: index/8 + boffset.
        "mark page as free in bittable"
        ch ← bitinst next.
        [ch allmask: (m ← 0200 lshift: 0-(index land: 7))⇒ [
            bitinst last ← ch xor: m]
        "already free?"].
        junkfile dskio: dadr
            command: CRR
            page: 0
            pages: 1
            onerror° [self error: ''dealloc'']
        dadr ← junkfile nextp.
        junkfile zaplabel.
        junkfile docommand: CWW]
    bitinst flush]'!

Directory understands: 'free | npages tpages ch i [
    self reset.
    "bitinst settopage: 1 char: 2.
    tpages ← bitinst nextword * 24."
    tpages ← 4872.
    npages ← 0.
    bitinst settopage: 1 char: boffset.
    for° i from: (1 to: tpages by: 8) do° [
        ch ← bitinst next.
        ch = 0377⇒ [npages ← npages+8]
        until° ch = 0 do° [
            [ch allmask: 1⇒ [npages ← npages+1]]
            ch ← ch lshift: ¯1]]
    ↑tpages - npages]'!

Directory understands: 'getNewSn | sn [
    bitinst settopage: 1 char: 010.
    sn ← bitinst into: (String new: 4).
    sn word: 2 ← (sn word: 2) + 1.
    [(sn word: 2) = 0⇒ [sn word: 1 ← (sn word: 1) + 1]].
    bitinst appendlast: sn.
    bitinst flush.
    ↑sn]'!

Directory understands: 'use [defdir ← self]'!
Directory understands: 'on: dirinst []'!

"dp0 ← Directory new use on: 0!"
```