111: PreludeIO の実装

↑ up

- issued: 2020-10-11
- 分類： B 機能追加
- status: Open

概要

PreludeIO を実装する。

2020-10-11 時点の残項目は以下の通り。入力、出力ともに疎通しているので、大物は例外周りであるといえる。

項番	項目	タスク数	実施済	備考
1	type FilePath	1		
2	data IOError, instance {Show|Eq} Error	3		
3	ioError	1		
4	userError	1		
5	catch	1		
6	putChar	1		
7	putStr	1		
8	getContents	1		
9	interact	1		
10	readFile	1		
11	writeFile	1		
12	appenFile	1		
13	readIO	1		
14	readLn	1		
		16	0	
			0.00%	
		残り	16	
			100.00%	

調査ログ

2021-10-28 (Thu)

putChar を追加する。

まず、lib/Prelude.hs に対する追加。これは、これで OK のはず：

```
$ git diff lib/Prelude.hs
diff --git a/compiler/lib/Prelude.hs b/compiler/lib/Prelude.hs
index 73dad25..d1f1ea2 100644
--- a/compiler/lib/Prelude.hs
+++ b/compiler/lib/Prelude.hs
@@ -1186,6 +1186,10 @@ instance (Read a) => Read [a] where
   readsPrec p = readList

 -- PreludeIO
```

```
+
+putChar :: Char -> IO ()
+putChar = Prim.putChar
+
 putStrLn :: String -> IO ()
 putStrLn =  Prim.putStrLn
```

つぎに、PreDefined.hs への追加。これも、これでいいはず：

```
$ git diff src/PreDefined.hs
diff --git a/compiler/src/PreDefined.hs b/compiler/src/PreDefined.hs
index 27db528..b104be6 100644
--- a/compiler/src/PreDefined.hs
+++ b/compiler/src/PreDefined.hs
@@ -330,6 +330,10 @@ primFloatAtanhCfun =  "Prim.floatAtanh" :>: Forall [] ([] :=> (tFloat `fn` tFlo
 primFloatShowCfun :: Assump
 primFloatShowCfun  = "Prim.floatShow" :>: Forall [] ([] :=> (tFloat `fn` tString))

+primPutCharCfun :: Assump
+primPutCharCfun =
+  "Prim.putChar" :>: Forall [] ([] :=> (tChar `fn` TAp tIO tUnit))
+
 primPutStrLnCfun :: Assump
 primPutStrLnCfun =
   "Prim.putStrLn" :>: Forall [] ([] :=> (TAp tList tChar `fn` TAp tIO tUnit))
@@ -387,6 +391,7 @@ primConsMems  = [ unitCfun, nilCfun, consCfun, pairCfun, tripleCfun
                 , errorCfun
                 , eFAILCfun
                 , primSeqCfun
+                , primPutCharCfun
                 , primPutStrLnCfun
                 , primGetCharCfun
                 , overloadedCfun
@@ -485,6 +490,7 @@ primNames  = fromList (primConsNames ++
                       , ("Prim.bindIO", "Prim.bindIO")
                       , ("Prim.failIO", "Prim.failIO")
                       , ("Prim.seq", "Prim.seq")
+                      , ("Prim.putChar", "Prim.putChar")
                       , ("Prim.putStrLn", "Prim.putStrLn")
```

```
                        , ("Prim.getChar", "Prim.getChar")
                        ])
```

さらに、runtime への追加だが、これは、今日は中途半端：

```
$ git diff ../brt/src/jp/ne/sakura/uhideyuki/brt/runtime
diff --git a/brt/src/jp/ne/sakura/uhideyuki/brt/runtime/Prim.java b/brt/src/jp/ne/sakura/uhideyuki/b
index 2a33576..0ebc446 100644
--- a/brt/src/jp/ne/sakura/uhideyuki/brt/runtime/Prim.java
+++ b/brt/src/jp/ne/sakura/uhideyuki/brt/runtime/Prim.java
@@ -19,6 +19,10 @@ public class Prim {
        return RTLib.seq;
     }

+    public static Expr mkputChar(){
+       return RTLib.putChar;
+    }
+
     public static Expr mkputStrLn(){
        return RTLib.putStrLn;
     }
diff --git a/brt/src/jp/ne/sakura/uhideyuki/brt/runtime/RTLib.java b/brt/src/jp/ne/sakura/uhideyuki/
index 064c005..e6d0247 100644
--- a/brt/src/jp/ne/sakura/uhideyuki/brt/runtime/RTLib.java
+++ b/brt/src/jp/ne/sakura/uhideyuki/brt/runtime/RTLib.java
@@ -22,6 +22,15 @@ class SeqFunc implements LambdaForm {
     }
 }

+class PutCharFunc implements LambdaForm {
+    public int arity(){ return 1; }
+    public Expr call(AtomExpr[] args){
+       assert args.length == arity();
+       IOWrapper.println("#### putChar is not implemented yet. ####");
+       return RTLib.app(Prim.mkretIO(), RTLib.unit);
+    }
+}
+
 class PutStrLnFunc implements LambdaForm {
```

```
     public int arity(){ return 1; }
     public Expr call(AtomExpr[] args){
@@ -175,6 +184,8 @@ public class RTLib {

     public static Expr seq = mkFun(new SeqFunc());

+    public static Expr putChar = mkFun(new PutCharFunc());
+
     public static Expr putStrLn = mkFun(new PutStrLnFunc());

     public static Expr fromChar(int c){ return mkLitChar(c); }
```

RTLib.putCharFunc の中身がダミーなので、putChar の使用は期待通りに動かない。

```
unno@unno-FMVD70GN7G ~/work/bissues/111
$ ~/prj/bunny/compiler/bin/bunny testrun t111putchar.hs
/home/unno/prj/bunny/compiler/bin/bunnyc -d ./jout/t111putchar --xno-implicit-prelude /home/unno/prj
/home/unno/prj/bunny/compiler/bin/bunnyc -d ./jout/t111putchar --xlibrary-path /home/unno/prj/bunny/
#### putChar is not implemented yet. ####
#### putChar is not implemented yet. ####
```

いちおう、make check が通る状態はキープしてあるので、この途中経過でコミットして、続きは後日。

IO は IOWrapper を介し、コマンドライン実行 runtime と Android ランタイムの両対応が必要な点に注意。

2021-10-29 (Fri)

putChar

RTLib.PutCharFunc の中身がダミーだったのを実装、また、これにあわせて IOWrapper.java, IOWapper.java2, MainIntentService も修正、putChar が疎通するようになった。

テストとして sample331.hs を追加した。

putStr

putChar ができたので、lib/Prelude.hs に以下を追加：

```
putStr :: String -> IO ()
putStr s = mapM_ putChar s
```

期待通り動作した：

```
unno@unno-FMVD70GN7G ~/work/bissues/111
$ cat t111putstr.hs
main = do
  putStr "Haskell"
  putChar ' '
  putStrLn "Compiler"
unno@unno-FMVD70GN7G ~/work/bissues/111
$ runhaskell t111putstr.hs
Haskell Compiler
unno@unno-FMVD70GN7G ~/work/bissues/111
$ ~/prj/bunny/compiler/bin/bunny testrun t111putstr.hs
/home/unno/prj/bunny/compiler/bin/bunnyc -d ./jout/t111putstr --xno-implicit-prelude /home/unno/prj/
/home/unno/prj/bunny/compiler/bin/bunnyc -d ./jout/t111putstr --xlibrary-path /home/unno/prj/bunny/c
Haskell Compiler
```

これを sample332.hs とする。

2021-11-07 (Sun)

例外をあつかう小さなサンプルを書いてみる。
GHC では Prelude に catch が含まれないようで、動かない。Hugs では動く。

```
unno@unno-FMVD70GN7G ~/work/bissues/111
$ cat catch_sample1b.hs
main :: IO ()
main = (ioError (userError "Error sample"))
  `catch`
  (\e -> print e)
unno@unno-FMVD70GN7G ~/work/bissues/111
$ runhaskell catch_sample1b.hs

catch_sample1b.hs:3:3: error:
    Variable not in scope: catch :: IO a0 -> (a1 -> IO ()) -> IO ()
  |
3 |   `catch`
  |   ^^^^^^^
```

```
unno@unno-FMVD70GN7G ~/work/bissues/111
$ runhugs catch_sample1b.hs
user error (Error sample)
```

GHC で動かすには、Control.Exception をインクルードしたうえで、例外ハンドラ関数の型を指定する必要があるようだ：

```
unno@unno-FMVD70GN7G ~/work/bissues/111
$ cat catch_sample1.hs
import           Control.Exception

main :: IO ()
main = (ioError (userError "Error sample"))
  `catch`
  ((\e -> print e) :: IOException -> IO ())
unno@unno-FMVD70GN7G ~/work/bissues/111
$ runhaskell catch_sample1.hs
user error (Error sample)
```

Bunny は Haskell 2010 に従うつもりなので、この件については Hugs で動作確認していくしかないかな。