```
--file Utils.mesa
--last modified by Bruce on  July 30, 1978  4:27 PM

DIRECTORY
  AltoDefs: FROM "AltoDefs" USING [PageSize],
  FSPDefs: FROM "FSPDefs",
  GPsortDefs:  FROM "gpsortdefs",
  InlineDefs: FROM "InlineDefs" USING [COPY],
  SegmentDefs:  FROM "segmentdefs",
  StreamDefs:  FROM "streamdefs" USING [DiskHandle, WriteBlock, ReadBlock],
  SystemDefs: FROM "SystemDefs" USING [AllocateResidentPages, FreePages];

Utils: PROGRAM
  IMPORTS SegmentDefs, StreamDefs, SystemDefs, FSPDefs
  EXPORTS GPsortDefs =
PUBLIC BEGIN OPEN GPsortDefs;

sortZone: FSPDefs.ZonePointer ← NIL;

Alloc: PROCEDURE [nwords: CARDINAL] RETURNS [p: POINTER] =
  BEGIN RETURN[FSPDefs.MakeNode[sortZone, nwords]] END;

Free: PROCEDURE [p: POINTER] = BEGIN FSPDefs.FreeNode[sortZone,p] END;

InitHeap: PROCEDURE[npages: CARDINAL] =
  BEGIN OPEN SystemDefs;
  IF sortZone # NIL THEN ERROR;  -- was EraseHeap[]
  sortZone ← FSPDefs.MakeNewZone[
    AllocateResidentPages[npages], npages*AltoDefs.PageSize, FreePages];
  END;

EraseHeap: PROCEDURE =
  BEGIN
  FSPDefs.DestroyZone[sortZone];
  sortZone ← NIL;
  END;

DeleteFile: PROCEDURE[dh: StreamDefs.DiskHandle] =
  BEGIN OPEN SegmentDefs;
  fh: FileHandle ← dh.file;
  LockFile[fh];
  dh.destroy[dh];
  UnlockFile[fh];
  DestroyFile[fh];
  END;

FillBuffer: PROCEDURE[file: FdHandle, size: INTEGER] =
  BEGIN OPEN file;
  cnt: CARDINAL;
  tail ← tail-head;
  IF tail > 0 THEN
    InlineDefs.COPY[@buffer↑[head],tail,buffer];  -- from, nwords, to
  cnt ← StreamDefs.ReadBlock[dh,@buffer↑[tail],size-tail];
  head ← 0;
  tail ← tail+cnt;
  RETURN;
  END;

FlushBuffer: PROCEDURE[file: FdHandle] =
  BEGIN OPEN file;
  [] ← StreamDefs.WriteBlock[dh, buffer, tail];
  tail ← 0;
  END;

END.
```