OS ノード 0 の認識メモリ領域の拡張について

2016/9/29

木下 直樹

1 はじめに

本研究のヘルスチェック機能では監視用 OS がサービス用 OS のカーネルテキスト部を監視する．このため，監視用 OS はサービス用 OS のカーネルテキスト部を取得できなければならない．すなわち，ある OS ノードが他 OS ノードのカーネルテキスト部を取得できなければならない．しかし，Mint では各 OS ノードが互いのカーネルテキスト部を認識できない．これは Mint で各 OS ノードの認識メモリ領域が重ならないように設計されているためである．このため，監視用 OS となる OS ノードの認識メモリ領域を拡張し，その OS ノードから他 OS ノードのカーネルテキスト部を取得できるようにする．ここで，本研究では OS ノード 0 を監視用 OS とする．

本資料では，OS ノード 0 の認識メモリ領域の拡張について述べる．また，以降で記載する認識メモリ領域とは，0x20000000 から 0x68000000 までのメモリ領域の内，指定した OS ノードが認識できる領域を指す．

2 OS ノード 0 の認識メモリ領域の拡張

2.1 概要

各 OS ノードのカーネルテキスト部は 0x20000000 から 0x68000000 までのメモリ領域に配置される．Kexec-Mint の初期設定により，この領域は各 OS ノードに等分配され，各 OS ノードは自身に分配された領域のみ認識できる．Kexec-Mint における各 OS ノードの認識メモリ領域を図 1 に示す．図 1 の OS ノード 0 の認識メモリ領域を拡張し，OS ノード 0 が図 1 で示すメモリ領域全域を認識することを目標とする．

2.2 方法

各 OS ノードの起動には OS ノード毎に異なる Kexec-Mint のソースコードを用い，各 OS ノードが認識できるメモリ領域はその OS ノードの起動に用いる Kexec-Mint の `kexec/firmware_memmap.c` ファイルに依存する．`kexec/firmware_memmap.c` ファイルでは，認識メモリ領域を先頭アドレスとサイズで定める．また，OS ノード 0 の認識メモリ領域は先頭アドレスが 0x20000000 である．このため，OS ノード 0 は自身の認識メモリ領域のサイズを大きくすることで 0x20000000 から 0x68000000 の領域を認識できる．これにより，OS ノード 0 が他 OS ノードのカーネルテキスト部を取得できるようにする．OS ノード 0 の認識メモリ領域のサイズを大きくし，各 OS ノードの認識メモリ領域を図 2 の状態にする．

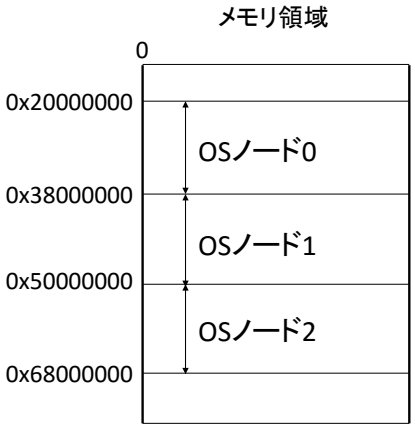

図 1　Kexec-Mint のソースコード改変前における各 OS ノードの認識メモリ領域

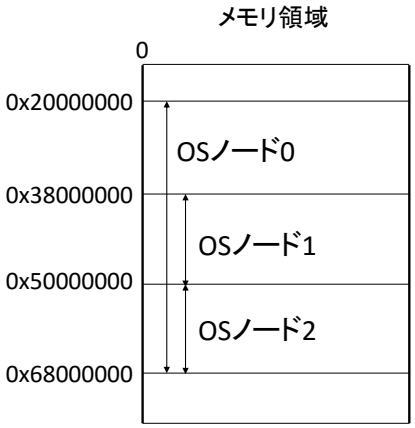

図 2　Kexec-Mint のソースコード改変後における各 OS ノードの認識メモリ領域

2.3　OS ノード 0 の認識メモリ領域の拡張結果

Kexec-Mint のソースコード改変後，Kexec-Mint の実行による出力結果から，OS ノード 0 の認識メモリ領域が目標の領域まで拡張できたことを確認した．しかし，この状態で OS ノード 2 を起動しようとすると計算機が強制的に再起動した．この問題の原因は特定できていない．ここで，拡張する領域サイズを小さくし，再度他 OS ノードの起動を試した．OS ノード 2 のカーネルテキスト部の配置領域を調査し，OS ノード 0 の認識領域をその領域まで拡張した．これにより，OS ノード 0 は走行する全 OS ノードのカーネルテキスト部を取得できるか否か調査した．OS ノード 2 の`/proc/iomem` ファイルから，OS ノード 2 のカーネルテキスト部は 0x60000000 までの領域に配置されることがわかった．このため，OS ノード 0 の認識メモリ領域を 0x60000000 まで拡張し，各 OS ノードの起動を試した．結果，全 OS ノードは起動できた．しかし，各 OS ノード起動後，OS ノード 2 でログインすると OS ノード 0 が停止する問題が発生した．この問題の原因も特定できていない．

2.4 kexec/firmware_memmap.c の改変箇所

OS ノード 0 を起動する Kexec-Mint について，kexec/firmware_memmap.c ファイルの改変箇所を以下に示し，説明する．削除行は行頭に − を付与し，追加行は行頭に + を付与する．

```
  280 /* Modify following three params to setup memmap */
  281
  282 mint_region1_start = (mint_memory_region % 5) * 0x20000;
  283 mint_region1_size  = 0x20000;
  284 mint_region2_start = 0x8000000 + 0x8000000*(mint_memory_region % 5);
  285 mint_region2_size  = 0x8000000;
  286 mint_region3_start = mint_memory_start;
- 287 mint_region3_size  = mint_memory_size - 0x8000000;
+ 287 mint_region3_size  = 0x48000000;
```

0x20000000 から 0x68000000 のメモリ領域において，各 OS ノードは 286，287 行目で定義するメモリ領域を認識できる．286 行目の変数 mint_memory_start は (0x20000000+[OSID]∗0x18000000) が格納されており，ここで認識メモリ領域の先頭アドレスを定義する．287 行目の変数 mint_memory_size には 0x20000000 が格納されおり，ここで認識メモリ領域のサイズを定義する．各変数は kexec/mint.h ファイルで定義されている．287 行目を改変し，OS ノード 0 の認識メモリ領域を拡張した．

3 今後の課題

今後の課題について以下で述べる．

(1) 監視用 OS からサービス用 OS のカーネルテキスト部のサイズを取得する機能の実現
各 OS ノードのカーネルテキスト部の先頭アドレスは固定であるが，サイズは OS 依存である．現状，このサイズは各 OS ノード自身の/proc/iomem ファイルから取得している．この方法では監視用 OS がサービス用 OS の支援無しにヘルスチェック機能を実現できない．このため，監視用 OS からサービス用 OS のカーネルテキスト部のサイズを取得する機能を実現する．

(2) OS ノード 0 の認識メモリ領域の拡張により，全 OS ノードが走行できなくなる原因の調査
本資料で，OS ノード 0 の認識メモリ領域を拡張した．しかし，この状態で全 OS ノードを走行させることはできなかった．この原因を調査し，走行する各 OS ノードのカーネルテキスト部を OS ノード 0 が取得できるようにする．

(3) 拡張したメモリ領域の読み込み専用領域化の実現
本資料で拡張したメモリ領域は書き込みに対する保護がされていない．これにより，現状，OS ノード 0 は拡張したメモリ領域に書き込みできる．このため，本来 OS ノード 1 または OS ノード 2 のみが使用すべきメモリ領域を OS ノード 0 よって改変されることが考えられる．今後は，

拡張したメモリ領域を読み込み専用化させる.

4 おわりに

本資料では，OS ノード 0 の認識メモリ領域の拡張について述べた.
今後は 3 章に挙げた課題に取り組む.