リスト1（コンテナ設定ファイル（conf/step1）の内容）

```
     1  # ステップ1で作成するコンテナ環境の定義ファイル
     2  # 名前空間を分離してbashを起動する
     3
     4  # /bin/bashを起動する
     5  PROG=/bin/bash
```

リスト2［コンテナ環境作成コマンド（create_mycont.sh）］

```
     1  #!/bin/bash
     2  #
     3  # コンテナ環境を作成するコマンド
     4
     5  if [ "$#" -lt 1 ]; then
     6      echo "Usage: $0 コンテナ名"
     7      exit 1
     8  fi
     9  MY_CONT=$1
    10
    11  echo "Create Container: $MY_CONT"
    12
    13  # 環境情報読み込み
    14  . $MYCON_TOP/MYENV
    15
    16  # コンテナ情報読み込み
    17  . $MYCON_TOP/conf/$MY_CONT
    18  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    19  ROOT_DIR=$CONT_DIR/root
    20
    21  # コンテナ環境用ディレクトリ作成
    22  mkdir -p $CONT_DIR/root
    23
    24  # コンテナ用のcgroupを作成
    25  sudo mkdir -p /sys/fs/cgroup/$MY_CONT
    26
    27  # モニタ処理プロセス用のコマンド・ファイル生成
    28  cat <<EOF > $ROOT_DIR/${MY_CONT}.mon
    29  #!/bin/bash
    30  echo "start ${MY_CONT}.mon ..."
    31  while [ -e $ROOT_DIR/${MY_CONT}.run ]
    32  do
    33    ip addr
    34    sleep 10
    35  done
    36  echo ".... finish"
    37  EOF
    38
    39  chmod a+x $ROOT_DIR/${MY_CONT}.mon
```

リスト3［コンテナの起動コマンド（start_mycont.sh）］

```
     1  #!/bin/bash
     2  #
     3  # コンテナを起動するコマンド
     4  #
     5
     6  if [ "$#" -lt 1 ]; then
     7      echo "Usage: $0 コンテナ名"
     8      exit 1
     9  fi
    10  MY_CONT=$1
    11
    12  echo "Start Container: $MY_CONT"
    13
    14  # 環境情報読み込み
    15  . $MYCON_TOP/MYENV
    16
    17  # コンテナ情報読み込み
    18  . $MYCON_TOP/conf/$MY_CONT
    19  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    20  ROOT_DIR=$CONT_DIR/root
    21
    22  # コンテナ環境のディレクトリへ移動
    23  cd $MYCON_TOP/work/$MY_CONT
    24
    25  # モニタ・スクリプトの実行制御を行うファイルを作成する（このファイルがある間モニタが動作する）
    26  touch $ROOT_DIR/${MY_CONT}.run
    27
    28  # 名前空間を分けてモニタ・スクリプトを起動する
    29  # リソース制御のためCGROUP名前空間は分離しない
    30  nohup unshare -i -m -n -p -u -U -T -f $ROOT_DIR/${MY_CONT}.mon &
    31  sleep 1
    32
    33  # モニタ・プロセスのPIDを取得する
```

```
                                                          S4_list.txt
    34  MON_PID=`$MYCON_TOP/env/my_contmon_pid $MY_CONT`
    35  echo "MON_PID = $MON_PID"
    36
    37  # 指定コマンドをモニタ・プロセスと同じ名前空間で実行する
    38  nsenter -t $MON_PID -m -u -i -n -p -U -T  --preserve-credentials -r -w $PROG
```

リスト4 [起動中のコンテナを停止るコマンド（stop_mycont.sh）]

```
     1  #!/bin/bash
     2  #
     3  # コンテナを停止するコマンド
     4
     5
     6  if [ "$#" -lt 1 ]; then
     7      echo "Usage: $0 コンテナ名"
     8      exit 1
     9  fi
    10  MY_CONT=$1
    11
    12  echo "Stop Container: $MY_CONT"
    13
    14  # 環境情報読み込み
    15  . $MYCON_TOP/MYENV
    16
    17  # コンテナ情報読み込み
    18  . $MYCON_TOP/conf/$MY_CONT
    19  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    20  ROOT_DIR=$CONT_DIR/root
    21
    22  echo "Remove MyCont -- $MY_CONT"
    23
    24  # コンテナで実行しているコマンドのPIDを求める
    25  PID=`$MYCON_TOP/env/my_cont_pid $MY_CONT`
    26  echo "PROG PID = $PID"
    27
    28  # コマンドを終了させる
    29  sudo kill -9 $PID
    30
    31  # モニタ・スクリプトの実行制御を行うファイルを削除する
    32  rm -f $ROOT_DIR/${MY_CONT}.run
```

リスト5 [コンテナ環境削除コマンド（remove_mycont.sh）]

```
     1  #!/bin/bash
     2  #
     3  # コンテナ環境を削除するコマンド
     4
     5  if [ "$#" -lt 1 ]; then
     6      echo "Usage: $0 コンテナ名"
     7      exit 1
     8  fi
     9  MY_CONT=$1
    10
    11  echo "Remove Container: $MY_CONT"
    12
    13  # 環境情報読み込み
    14  . $MYCON_TOP/MYENV
    15
    16  # コンテナ情報読み込み
    17  . $MYCON_TOP/conf/$MY_CONT
    18  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    19  ROOT_DIR=$CONT_DIR/root
    20
    21  # コンテナ用cgroupを削除 ★エラーになるので無効化
    22  #sudo rm -rf  /sys/fs/cgroup/$MY_CONT
    23
    24  # コンテナ環境用ディレクトリ削除
    25  sudo -E rm -rf $MYCON_TOP/work/$MY_CONT
```

リスト6 [コンテナ・モニタ・プロセスのPIDを取得するコマンド（env/my_contmon_pid）]

```
     1  #!/bin/bash
     2  #
     3  # コンテナのモニタ・プロセスのPIDを取得するコマンド
     4
     5  if [ "$#" -lt 1 ]; then
     6      echo "Usage: $0 コンテナ名"
     7      exit 1
     8  fi
     9  MY_CONT=$1
    10
    11  # 環境情報読み込み
    12  . $MYCON_TOP/MYENV
```

```
    13
    14  # コンテナ情報読み込み
    15  . $MYCON_TOP/conf/$MY_CONT
    16
    17  # モニタ・プロセスのプロセスIDを取得する
    18  EXE=${MY_CONT}.mon
    19  PID=`ps -aex | grep PWD=$MYCON_TOP/work/$MY_CONT |  awk '{print($1,$5,$6)}' | grep $EXE | grep -v grep | cut -d ' '
-f 1`
    20
    21  echo $PID
```

リスト7 [コマンドのPIDを取得するコマンド（env/my_cont_pid）]

```
     1  #!/bin/bash
     2  #
     3  # コンテナで動作しているプログラムのPIDを取得するコマンド
     4
     5  if [ "$#" -lt 1 ]; then
     6      echo "Usage: $0 コンテナ名"
     7      exit 1
     8  fi
     9  MY_CONT=$1
    10
    11  # 環境情報読み込み
    12  . $MYCON_TOP/MYENV
    13
    14  # コンテナ情報読み込み
    15  . $MYCON_TOP/conf/$MY_CONT
    16
    17  # 実行プログラムのPIDを取得する
    18  MON=${MY_CONT}.mon
    19  EXE=`echo $PROG | cut -d ' ' -f 1`
    20  PID=`ps -aex | grep PWD=$MYCON_TOP/work/$MY_CONT |  awk '{print($1,$5,$6)}' | grep -v $MON | grep $EXE | cut -d ' '
-f 1`
    21
    22  echo $PID
```

リスト8 [修正後のコンテナ環境作成コマンド（create_mycont.sh）]

```
     1  #!/bin/bash
     2  #
     3  # コンテナ環境を作成するコマンド
     4
     5  if [ "$#" -lt 1 ]; then
     6      echo "Usage: $0 コンテナ名"
     7      exit 1
     8  fi
     9  MY_CONT=$1
    10
    11  echo "Create Container: $MY_CONT"
    12
    13  # 環境情報読み込み
    14  . $MYCON_TOP/MYENV
    15
    16  # コンテナ情報読み込み
    17  . $MYCON_TOP/conf/$MY_CONT
    18  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    19  ROOT_DIR=$CONT_DIR/root
    20
    21  # コンテナ環境用ディレクトリ作成
    22  mkdir -p $CONT_DIR/root
    23  mkdir -p $CONT_DIR/upper
    24  mkdir -p $CONT_DIR/work
    25
    26  # overlay FSCONT_DIR
    27  # システムの/をベースとして利用する -> Raspberry Pi OSのアプリを使える
    28  sudo -E  mount -t overlay $MY_CONT -o lowerdir=/,upperdir=$CONT_DIR/upper,workdir=$CONT_DIR/work $CONT_DIR/root
    29
    30  # procファイル・システムは共有する(bind)
    31  mkdir -p $ROOT_DIR/proc
    32  sudo -E mount --bind /proc $ROOT_DIR/proc
    33
    34  # コンテナ用のcgroupを作成
    35  sudo mkdir -p /sys/fs/cgroup/$MY_CONT
    36
    37  # モニタ処理プロセス用のコマンド・ファイル生成
    38  cat <<EOF > $ROOT_DIR/${MY_CONT}.mon
    39  #!/bin/bash
    40  echo "start ${MY_CONT}.mon ..."
    41  while [ -e /${MY_CONT}.run ]
    42  do
    43    ip addr
    44    sleep 10
    45  done
    46  echo ".... finish"
    47  EOF
```

```
    48
    49  chmod a+x $ROOT_DIR/${MY_CONT}.mon
```

リスト9 [修正後のコンテナ環境起動コマンド（start_mycont.sh）]

```
     1  #!/bin/bash
     2  #
     3  # コンテナを起動するコマンド
     4
     5  if [ "$#" -lt 1 ]; then
     6      echo "Usage: $0 コンテナ名"
     7      exit 1
     8  fi
     9  MY_CONT=$1
    10
    11  echo "Start Container: $MY_CONT"
    12
    13  # 環境情報読み込み
    14  . $MYCON_TOP/MYENV
    15
    16  # コンテナ情報読み込み
    17  . $MYCON_TOP/conf/$MY_CONT
    18  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    19  ROOT_DIR=$CONT_DIR/root
    20
    21  # コンテナ環境のディレクトリへ移動
    22  cd $MYCON_TOP/work/$MY_CONT
    23
    24  # モニタ・スクリプトの実行制御を行うファイルを作成する（このファイルがある間モニタが動作する）
    25  touch $ROOT_DIR/${MY_CONT}.run
    26
    27  # 名前空間を分けてモニタ・スクリプトを起動する
    28  # リソース制御のためCGROUP名前空間は分離しない
    29  nohup unshare -i -m -n -p -u -U -T -R `pwd`/root -f /${MY_CONT}.mon &
    30  sleep 1
    31
    32  # モニタ・プロセスのPIDを取得する
    33  MON_PID=`$MYCON_TOP/env/my_contmon_pid $MY_CONT`
    34  echo "MON_PID = $MON_PID"
    35
    36  # 指定コマンドをモニタ・プロセスと同じ名前空間で実行する
    37  nsenter -t $MON_PID -m -u -i -n -p -U -T  --preserve-credentials -r -w $PROG
```

リスト10 [修正後のコンテナ環境削除コマンド（remove_mycont.sh）]

```
     1  #!/bin/bash
     2  #
     3  # コンテナ環境を削除するコマンド
     4
     5  if [ "$#" -lt 1 ]; then
     6      echo "Usage: $0 コンテナ名"
     7      exit 1
     8  fi
     9  MY_CONT=$1
    10
    11  echo "Remove Container: $MY_CONT"
    12
    13  # 環境情報読み込み
    14  . $MYCON_TOP/MYENV
    15
    16  # コンテナ情報読み込み
    17  . $MYCON_TOP/conf/$MY_CONT
    18  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    19  ROOT_DIR=$CONT_DIR/root
    20
    21  # コンテナ用cgroupを削除 ★エラーになるので無効化
    22  #sudo rm -rf  /sys/fs/cgroup/$MY_CONT
    23
    24  # procファイル・システムの共有を止める
    25  sudo -E umount $ROOT_DIR/proc
    26
    27  # overlay FS
    28  sudo -E umount $MY_CONT
    29
    30  # コンテナ環境用ディレクトリ削除
    31  sudo -E rm -rf $MYCON_TOP/work/$MY_CONT
```

リスト11（設定フィルタにネットワーク関連の設定を追加）

```
     1  # ステップ3で作成するコンテナ環境の定義ファイル
     2  # 名前空間を分離してbashを起動する
     3
     4  # /bin/bashを起動する
     5  PROG=/bin/bash
     6
```

```
 7  # ネットワークの設定
 8  NET_NAME=my_net
 9  BRIDGE_NAME=my_net_br
10  BRIDGE_IP=10.1.1.1
11  IP=10.1.1.100
12  NETWORK_BITS=24
13  BROADCAST=10.1.1.255
```

リスト12 [修正後のコンテナ環境作成コマンド (create_mycont.sh)]

```
 1  #!/bin/bash
 2  #
 3  # コンテナ環境を作成するコマンド
 4
 5  if [ "$#" -lt 1 ]; then
 6      echo "Usage: $0 コンテナ名"
 7      exit 1
 8  fi
 9  MY_CONT=$1
10
11  echo "Create Container: $MY_CONT"
12
13  # 環境情報読み込み
14  . $MYCON_TOP/MYENV
15
16  # コンテナ情報読み込み
17  . $MYCON_TOP/conf/$MY_CONT
18  CONT_DIR=$MYCON_TOP/work/$MY_CONT
19  ROOT_DIR=$CONT_DIR/root
20
21  # コンテナ環境用ディレクトリ作成
22  mkdir -p $CONT_DIR/root
23  mkdir -p $CONT_DIR/upper
24  mkdir -p $CONT_DIR/work
25
26  # overlay FSCONT_DIR
27  # システムの/をベースとして利用する -> Raspberry Pi OSのアプリを使える
28  sudo -E  mount -t overlay $MY_CONT -o lowerdir=/,upperdir=$CONT_DIR/upper,workdir=$CONT_DIR/work $CONT_DIR/root
29
30  # procファイル・システムは共有する(bind)
31  mkdir -p $ROOT_DIR/proc
32  sudo -E mount --bind /proc $ROOT_DIR/proc
33
34  # コンテナ用のcgroupを作成
35  sudo mkdir -p /sys/fs/cgroup/$MY_CONT
36
37  # 仮想ブリッジを作成
38  if [ $BRIDGE_NAME"x" != "x" ]; then
39      $MYCON_TOP/net/bridge_create.sh $BRIDGE_NAME
40  fi
41
42  # 仮想Ethernetデバイスを作成
43  if [ $NET_NAME"x" != "x" ]; then
44      $MYCON_TOP/net/veth_create.sh $NET_NAME
45  fi
46
47  # モニタ処理プロセス用のコマンド・ファイル生成
48  cat <<EOF > $ROOT_DIR/${MY_CONT}.mon
49  #!/bin/bash
50  echo "start ${MY_CONT}.mon ..."
51  while [ -e /${MY_CONT}.run ]
52  do
53    ip addr
54    sleep 10
55  done
56  echo ".... finish"
57  EOF
58
59  chmod a+x $ROOT_DIR/${MY_CONT}.mon
```

リスト13 [修正後のコンテナ環境起動コマンド (start_mycont.sh)]

```
 1  #!/bin/bash
 2  #
 3  # コンテナを起動するコマンド
 4  #
 5
 6  if [ "$#" -lt 1 ]; then
 7      echo "Usage: $0 コンテナ名"
 8      exit 1
 9  fi
10  MY_CONT=$1
11
12  echo "Start Container: $MY_CONT"
13
```

```
14  # 環境情報読み込み
15  . $MYCON_TOP/MYENV
16
17  # コンテナ情報読み込み
18  . $MYCON_TOP/conf/$MY_CONT
19  CONT_DIR=$MYCON_TOP/work/$MY_CONT
20  ROOT_DIR=$CONT_DIR/root
21
22  # コンテナ環境のディレクトリへ移動
23  cd $MYCON_TOP/work/$MY_CONT
24
25  # モニタ・スクリプトの実行制御を行うファイルを作成する（このファイルがある間モニタが動作する）
26  touch $ROOT_DIR/${MY_CONT}.run
27
28  # 名前空間を分けてモニタ・スクリプトを起動する
29  # リソース制御のためCGROUP名前空間は分離しない
30  nohup unshare -i -m -n -p -u -U -T -R `pwd`/root -f /${MY_CONT}.mon &
31  sleep 1
32
33  # モニタ・プロセスのPIDを取得する
34  MON_PID=`$MYCON_TO      P/env/my_contmon_pid $MY_CONT`
35  echo "MON_PID = $MON_PID"
36
37  # ネットワーク設定
38  if [ $NET_NAME"x" != "x" ]; then
39      $MYCON_TOP/net/net_attach.sh $MON_PID $NET_NAME $IP $BRIDGE_NAME $BRIDGE_IP $NETWORK_BITS $BROADCAST
40  fi
41
42  # 指定コマンドをモニタ・プロセスと同じ名前空間で実行する
43  # リソース制御のためCGROUP名前空間は分離しない
44  nsenter -t $MON_PID -m -u -i -n -p -U -T  --preserve-credentials -r -w $PROG
```

リスト14［修正後のコンテナ環境停止コマンド（stop_mycont.sh）］

```
 1  #!/bin/bash
 2  #
 3  # コンテナを停止するコマンド
 4
 5
 6  if [ "$#" -lt 1 ]; then
 7      echo "Usage: $0 コンテナ名"
 8      exit 1
 9  fi
10  MY_CONT=$1
11
12  echo "Stop Container: $MY_CONT"
13
14  # 環境情報読み込み
15  . $MYCON_TOP/MYENV
16
17  # コンテナ情報読み込み
18  . $MYCON_TOP/conf/$MY_CONT
19  CONT_DIR=$MYCON_TOP/work/$MY_CONT
20  ROOT_DIR=$CONT_DIR/root
21
22  echo "Remove MyCont -- $MY_CONT"
23
24  # コンテナで実行しているコマンドのPIDを求める
25  PID=`$MYCON_TOP/env/my_cont_pid $MY_CONT`
26  echo "PROG PID = $PID"
27
28  # コマンドを終了させる
29  sudo kill -9 $PID
30
31  # NET名前空間のリンク削除する
32  if [ $NET_NAME"x" != "x" ]; then
33      $MYCON_TOP/net/net_detach.sh $PID $NET_NAME $IP $BRIDGE_NAME $BRIDGE_IP $NETWORK_BITS $BROADCAST
34  fi
35
36  # モニタ・スクリプトの実行制御を行うファイルを削除する
37  rm -f $ROOT_DIR/${MY_CONT}.run
```

リスト15［修正後のコンテナ環境削除コマンド（remove_mycont.sh）］

```
 1  #!/bin/bash
 2  #
 3  # コンテナ環境を削除するコマンド
 4
 5  if [ "$#" -lt 1 ]; then
 6      echo "Usage: $0 コンテナ名"
 7      exit 1
 8  fi
 9  MY_CONT=$1
10
11  echo "Remove Container: $MY_CONT"
```

```
12
13  # 環境情報読み込み
14  . $MYCON_TOP/MYENV
15
16  # コンテナ情報読み込み
17  . $MYCON_TOP/conf/$MY_CONT
18  CONT_DIR=$MYCON_TOP/work/$MY_CONT
19  ROOT_DIR=$CONT_DIR/root
20
21  # 仮想Ethernetデバイスを削除
22  if [ $NET_NAME"x" != "x" ]; then
23      $MYCON_TOP/net/veth_remove.sh $NET_NAME
24  fi
25
26  # 仮想ブリッジ削除
27  if [ $BRIDGE_NAME"x" != "x" ]; then
28      $MYCON_TOP/net/bridge_remove.sh $BRIDGE_NAME
29  fi
30
31  # コンテナ用cgroupを削除 ★エラーになるので無効化
32  #sudo rm -rf  /sys/fs/cgroup/$MY_CONT
33
34  # procファイル・システムの共有を止める
35  sudo -E umount $ROOT_DIR/proc
36
37  # overlay FS
38  sudo -E umount $MY_CONT
39
40  # コンテナ環境用ディレクトリ削除
41  sudo -E rm -rf $MYCON_TOP/work/$MY_CONT
```

リスト16 [仮想ブリッジの作成コマンド（net/bridge_create.sh）]

```
 1  #!/bin/bash
 2  #
 3  # 仮想ブリッジを作成するコマンド
 4
 5  if [ "$#" -lt 1 ]; then
 6      echo "Usage: $0 ブリッジ名"
 7      exit 1
 8  fi
 9  BR_NAME=$1
10
11  echo "Create Bridge: $BR_NAME"
12
13  # 仮想ブリッジが存在しなければ作成する
14  ip link show $BR_NAME
15  if [ $? != 0 ]; then
16      sudo -E ip link add name $BR_NAME type bridge
17      sudo -E ip link set dev $BR_NAME up
18  fi
```

リスト17 [仮想ブリッジの削除マンド（net/bridge_remove.sh）]

```
 1  #!/bin/bash
 2  #
 3  # 仮想ブリッジを削除するコマンド
 4
 5  if [ "$#" -lt 1 ]; then
 6      echo "Usage: $0 ブリッジ名"
 7      exit 1
 8  fi
 9  BR_NAME=$1
10
11  echo "Remove Bridge: $BR_NAME"
12
13  # 仮想ブリッジに接続されているデバイスがなければ，仮想ブリッジを削除する
14  NUM=`sudo -E ip link show type bridge_slave | grep -w $BR_NAME | wc -l`
15  if [ $NUM = 0 ]; then
16      sudo -E ip link delete name $BR_NAME
17  fi
```

リスト18 [仮想Ethernetデバイスの作成（net/veth_create.sh）]

```
 1  #!/bin/bash
 2  #
 3  # 仮想Ethernetデバイスを作成するコマンド
 4
 5  if [ "$#" -lt 1 ]; then
 6      echo "Usage: $0 仮想Ethernetデバイス名"
 7      exit 1
 8  fi
 9  VETH_NAME=$1
10
11  echo "Create Virtual Ethernet Device: $VETH_NAME"
```

```
    12
    13  # 仮想Ethernetデバイスを作成する
    14  sudo -E ip link add name ${VETH_NAME}-eth type veth peer name ${VETH_NAME}-br
```

リスト19 [仮想Ethernetデバイスの削除コマンド (net/veth_remove.sh)]

```
     1  #!/bin/bash
     2  #
     3  # 仮想Ethernetデバイスを削除するコマンド
     4
     5  if [ "$#" -lt 1 ]; then
     6      echo "Usage: $0 仮想Ethernetデバイス名"
     7      exit 1
     8  fi
     9  VETH_NAME=$1
    10
    11  echo "Remove Virtual Ethernet Device: $VETH_NAME"
    12
    13  # 仮想Ethernetデバイスを削除する
    14  sudo -E ip link delete name ${VETH_NAME}-eth
```

リスト20 [ネットワーク接続コマンド (net/net_attach.sh)]

```
     1  #!/bin/bash
     2  #
     3  # コンテナにネットワークを接続するコマンド
     4
     5  if [ "$#" -lt 7 ]; then
     6      echo "Usage: $0 コンテナPID ネットワーク名 IPアドレス ブリッジ名 ブリッジIP ネットワークアドレス部 ブロードキャ
ストアドレス"
     7      exit 1
     8  fi
     9  PID=$1
    10  NET_NAME=$2
    11  IP=$3
    12  BRIDGE_NAME=$4
    13  BRIDGE_IP=$5
    14  NETWORK_BITS=$6
    15  BROADCAST=$7
    16
    17  # 環境情報の読み込み
    18  . $MYCON_TOP/MYENV
    19
    20  echo "Attach network: $PID $NET_NAME $IP $BRIDGE_NAME $BRIDGE_IP $NETWORK_BITS $BROADCAST"
    21
    22  # ipコマンドでNET名前空間を認識できるようにリンクを張る
    23  sudo -E mkdir -p /var/run/netns/
    24  sudo -E ln -sf /proc/$PID/ns/net /var/run/netns/$NET_NAME
    25
    26  # 仮想ブリッジに仮想Ethernetデバイスの一方を接続
    27  sudo -E ip addr add ${BRIDGE_IP}/${NETWORK_BITS} broadcast $BROADCAST label $BRIDGE_NAME dev $BRIDGE_NAME
    28  sudo -E ip link set dev ${NET_NAME}-br master $BRIDGE_NAME
    29  sudo -E ip link set dev ${NET_NAME}-br up
    30
    31  # 仮想Ethernetデバイスのもう一方をコンテナに接続
    32  sudo -E ip link set dev ${NET_NAME}-eth netns $NET_NAME
    33  sudo -E ip netns exec $NET_NAME ip addr add ${IP}/${NETWORK_BITS} dev ${NET_NAME}-eth
    34  sudo -E ip netns exec $NET_NAME ip link set dev ${NET_NAME}-eth up
    35  sudo -E ip netns exec $NET_NAME ip route add default via ${BRIDGE_IP}
    36  sudo -E ip netns exec $NET_NAME ip link set dev lo up
```

リスト21 [ネットワーク切断コマンド (net/net_detach.sh)]

```
     1  #!/bin/bash
     2  #
     3  # コンテナからネットワークを切断するコマンド
     4
     5
     6  if [ "$#" -lt 1 ]; then
     7      echo "Usage: $0 コンテナPID ネットワーク名 IPアドレス ブリッジ名 ブリッジIP ネットワークアドレス部 ブロードキャ
ストアドレス"
     8      exit 1
     9  fi
    10  PID=$1
    11  NET_NAME=$2
    12  IP=$3
    13  BRIDGE_NAME=$4
    14  BRIDGE_IP=$5
    15  NETWORK_BITS=$6
    16  BROADCAST=$7
    17
    18  echo "Detach network: $PID $NET_NAME $IP $BRIDGE_NAME $BRIDGE_IP $NETWORK_BITS $BROADCAST"
    19
    20  # 環境情報
    21  . $MYCON_TOP/MYENV
```

```
    22
    23  # 仮想Ethernetデバイスをコンテナから切断
    24  sudo -E ip netns exec $NET_NAME ip link set dev lo down
    25  sudo -E ip netns exec $NET_NAME ip route delete default via ${BRIDGE_IP}
    26  sudo -E ip netns exec $NET_NAME ip link set dev ${NET_NAME}-eth down
    27  sudo -E ip netns exec $NET_NAME ip addr delete ${IP}/${NETWORK_BITS} dev ${NET_NAME}-eth
    28  sudo -E ln -sf /proc/$$/ns/net /var/run/netns/host_ns
    29  sudo -E ip netns exec $NET_NAME ip link set dev ${NET_NAME}-eth netns host_ns
    30
    31  # NET名前空間のリンク削除する
    32  sudo -E rm -f /var/run/netns/$NET_NAME
```

リスト22［設定ファイルに共有フォルダの設定を追加（conf/step4）］

```
     1  # ステップ4で作成するコンテナ環境の定義ファイル
     2  # 名前空間を分離してbashを起動する
     3
     4  # /bin/bashを起動する
     5  PROG=/bin/bash
     6
     7  # ネットワークの設定
     8  NET_NAME=my_net
     9  BRIDGE_NAME=my_net_br
    10  BRIDGE_IP=10.1.1.1
    11  IP=10.1.1.100
    12  NETWORK_BITS=24
    13  BROADCAST=10.1.1.255
    14
    15  # ホストのディレクトリを/mywebにマウントして共有
    16  HOST_DIR=/home/cq/CQ/PROG/MyCON/work/sharedVol/web
    17  DIR=/myweb
```

リスト23［修正後のコンテナ環境作成コマンド（create_mycont.sh）］

```
     1  #!/bin/bash
     2  #
     3  # コンテナ環境を作成するコマンド
     4
     5  if [ "$#" -lt 1 ]; then
     6      echo "Usage: $0 コンテナ名"
     7      exit 1
     8  fi
     9  MY_CONT=$1
    10
    11  echo "Create Container: $MY_CONT"
    12
    13  # 環境情報読み込み
    14  . $MYCON_TOP/MYENV
    15
    16  # コンテナ情報読み込み
    17  . $MYCON_TOP/conf/$MY_CONT
    18  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    19  ROOT_DIR=$CONT_DIR/root
    20
    21  # コンテナ環境用ディレクトリ作成
    22  mkdir -p $CONT_DIR/root
    23  mkdir -p $CONT_DIR/upper
    24  mkdir -p $CONT_DIR/work
    25
    26  # overlay FSCONT_DIR
    27  # システムの/をベースとして利用する -> Raspberry Pi OSのアプリを使える
    28  sudo -E  mount -t overlay $MY_CONT -o lowerdir=/,upperdir=$CONT_DIR/upper,workdir=$CONT_DIR/work $CONT_DIR/root
    29
    30  # procファイル・システムは共有する(bind)
    31  mkdir -p $ROOT_DIR/proc
    32  sudo -E mount --bind /proc $ROOT_DIR/proc
    33
    34  # コンテナ用のcgroupを作成
    35  sudo mkdir -p /sys/fs/cgroup/$MY_CONT
    36
    37  # ホストとの共有ストレージをバインド・マウントで用意
    38  if [ $HOST_DIR"x" != "x" ]; then
    39      mkdir -p $CONT_DIR/root$DIR
    40      sudo -E mount --bind $HOST_DIR $CONT_DIR/root$DIR
    41  fi
    42
    43  # 仮想ブリッジを作成
    44  if [ $BRIDGE_NAME"x" != "x" ]; then
    45      $MYCON_TOP/net/bridge_create.sh $BRIDGE_NAME
    46  fi
    47
    48  # 仮想Ethernetデバイスを作成
    49  if [ $NET_NAME"x" != "x" ]; then
    50      $MYCON_TOP/net/veth_create.sh $NET_NAME
    51  fi
    52
```

```
 53  # モニタ処理プロセス用のコマンド・ファイル生成
 54  cat <<EOF > $ROOT_DIR/${MY_CONT}.mon
 55  #!/bin/bash
 56  echo "start ${MY_CONT}.mon ..."
 57  while [ -e /${MY_CONT}.run ]
 58  do
 59    ip addr
 60    sleep 10
 61  done
 62  echo ".... finish"
 63  EOF
 64
 65  chmod a+x $ROOT_DIR/${MY_CONT}.mon
```

リスト24 [修正後のコンテナ環境削除コマンド（step4_remove_mycont.sh）]

```
  1  #!/bin/bash
  2  #
  3  # コンテナ環境を削除するコマンド
  4
  5  if [ "$#" -lt 1 ]; then
  6      echo "Usage: $0 コンテナ名"
  7      exit 1
  8  fi
  9  MY_CONT=$1
 10
 11  echo "Remove Container: $MY_CONT"
 12
 13  # 環境情報読み込み
 14  . $MYCON_TOP/MYENV
 15
 16  # コンテナ情報読み込み
 17  . $MYCON_TOP/conf/$MY_CONT
 18  CONT_DIR=$MYCON_TOP/work/$MY_CONT
 19  ROOT_DIR=$CONT_DIR/root
 20
 21  # 仮想Ethernetデバイスを削除
 22  if [ $NET_NAME"x" != "x" ]; then
 23      $MYCON_TOP/net/veth_remove.sh $NET_NAME
 24  fi
 25
 26  # 仮想ブリッジ削除
 27  if [ $BRIDGE_NAME"x" != "x" ]; then
 28      $MYCON_TOP/net/bridge_remove.sh $BRIDGE_NAME
 29  fi
 30
 31  # ホストとの共有ストレージをアンマウント
 32  if [ $HOST_DIR"x" != "x" ]; then
 33      sudo -E umount $CONT_DIR/root$DIR
 34  fi
 35
 36  # コンテナ用cgroupを削除 ★エラーになるので無効化
 37  #sudo rm -rf  /sys/fs/cgroup/$MY_CONT
 38
 39  # procファイル・システムの共有を止める
 40  sudo -E umount $ROOT_DIR/proc
 41
 42  # overlay FS
 43  sudo -E umount $MY_CONT
 44
 45  # コンテナ環境用ディレクトリ削除
 46  sudo -E rm -rf $MYCON_TOP/work/$MY_CONT
```

リスト25 [設定ファイルにポートフォワードの設定を追加（conf/step5）]

```
  1  # ステップ5で作成するコンテナ環境の定義ファイル
  2
  3  # pythonでWEBサーバを起動する
  4  PROG='python -m http.server 8000 --directory /myweb'
  5
  6  # ネットワークの設定
  7  NET_NAME=c_web_net
  8  BRIDGE_NAME=c_web_br
  9  BRIDGE_IP=10.1.1.1
 10  IP=10.1.1.100
 11  NETWORK_BITS=24
 12  BROADCAST=10.1.1.255
 13
 14  # ホストのディレクトリを/mywebにマウントして共有
 15  HOST_DIR=/home/cq/CQ/PROG/MyCON/work/sharedVol/web
 16  DIR=/myweb
 17
 18  # ポートフォワード設定（8888->8000）
 19  HOST_PORT=8888
```

```
    20  PORT=8000
```

リスト26［修正後のコンテナ環境起動コマンド（start_mycont.sh）］

```
     1  #!/bin/bash
     2  #
     3  # コンテナを起動するコマンド
     4  #
     5
     6  if [ "$#" -lt 1 ]; then
     7      echo "Usage: $0 コンテナ名"
     8      exit 1
     9  fi
    10  MY_CONT=$1
    11
    12  echo "Start Container: $MY_CONT"
    13
    14  # 環境情報読み込み
    15  . $MYCON_TOP/MYENV
    16
    17  # コンテナ情報読み込み
    18  . $MYCON_TOP/conf/$MY_CONT
    19  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    20  ROOT_DIR=$CONT_DIR/root
    21
    22  # コンテナ環境のディレクトリへ移動
    23  cd $MYCON_TOP/work/$MY_CONT
    24
    25  # モニタ・スクリプトの実行制御を行うファイルを作成する（このファイルがある間モニタが動作する）
    26  touch $ROOT_DIR/${MY_CONT}.run
    27
    28  # 名前空間を分けてモニタ・スクリプトを起動する
    29  # リソース制御のためCGROUP名前空間は分離しない
    30  nohup unshare -i -m -n -p -u -U -T -R `pwd`/root -f /${MY_CONT}.mon &
    31  sleep 1
    32
    33  # モニタ・プロセスのPIDを取得する
    34  MON_PID=`$MYCON_TOP/env/my_contmon_pid $MY_CONT`
    35  echo "MON_PID = $MON_PID"
    36
    37  # ネットワーク設定
    38  if [ $NET_NAME"x" != "x" ]; then
    39      $MYCON_TOP/net/net_attach.sh $MON_PID $NET_NAME $IP $BRIDGE_NAME $BRIDGE_IP $NETWORK_BITS $BROADCAST
    40  fi
    41
    42  # ポートフォワードを設定する
    43  if [ $HOST_PORT"x" != "x" ]; then
    44      $MYCON_TOP/net/portfwd_create.sh $HOST_PORT $IP $PORT
    45  fi
    46
    47  # 指定コマンドをモニタ・プロセスと同じ名前空間で実行する
    48  nsenter -t $MON_PID -m -u -i -n -p -U -T  --preserve-credentials -r -w $PROG
```

リスト27［修正後のコンテナ環境停止コマンド（stop_mycont.sh）］

```
     1  #!/bin/bash
     2  #
     3  # コンテナを停止するコマンド
     4
     5
     6  if [ "$#" -lt 1 ]; then
     7      echo "Usage: $0 コンテナ名"
     8      exit 1
     9  fi
    10  MY_CONT=$1
    11
    12  echo "Stop Container: $MY_CONT"
    13
    14  # 環境情報読み込み
    15  . $MYCON_TOP/MYENV
    16
    17  # コンテナ情報読み込み
    18  . $MYCON_TOP/conf/$MY_CONT
    19  CONT_DIR=$MYCON_TOP/work/$MY_CONT
    20  ROOT_DIR=$CONT_DIR/root
    21
    22  echo "Remove MyCont -- $MY_CONT"
    23
    24  # コンテナで実行しているコマンドのPIDを求める
    25  PID=`$MYCON_TOP/env/my_cont_pid $MY_CONT`
    26  echo "PROG PID = $PID"
    27
    28  # コマンドを終了させる
    29  sudo kill -9 $PID
    30
```

```
31  # ポートフォワードの設定を解除する
32  if [ $HOST_PORT"x" != "x" ]; then
33      $MYCON_TOP/net/portfwd_remove.sh $HOST_PORT $PORT
34  fi
35
36  # NET名前空間のリンク削除する
37  if [ $NET_NAME"x" != "x" ]; then
38      $MYCON_TOP/net/net_detach.sh $PID $NET_NAME $IP $BRIDGE_NAME $BRIDGE_IP $NETWORK_BITS $BROADCAST
39  fi
40
41  # モニタ・スクリプトの実行制御を行うファイルを削除する
42  rm -f $ROOT_DIR/${MY_CONT}.run
```

リスト28 [ポートフォワード設定コマンド (net/portfwd_create.sh)]

```
 1  #!/bin/bash
 2  #
 3  # ポートフォワードの設定を行うコマンド
 4
 5  if [ "$#" -lt 3 ]; then
 6      echo "Usage: $0 ホストポート番号 IPアドレス ポート番号"
 7      exit 1
 8  fi
 9  HOST_PORT=$1
10  IP=$2
11  PORT=$3
12
13  echo "Create Port forward: $HOST_PORT $IP:$PORT"
14
15  # 環境情報の読み込み
16  . $MYCON_TOP/MYENV
17
18  # MYNATテーブルがなければ作成する
19  sudo -E nft list table MYNAT
20  if [ $? != 0 ]; then
21      sudo nft flush ruleset
22      sudo nft add table MYNAT
23      sudo nft 'add chain MYNAT postrouting { type nat hook postrouting priority 100 ; }'
24      sudo nft 'add chain MYNAT prerouting { type nat hook prerouting priority -100; }'
25      sudo nft add rule MYNAT postrouting masquerade
26  fi
27
28  # ポートフォワードの設定を行う
29  sudo -E nft add rule MYNAT prerouting tcp dport { $HOST_PORT } dnat ${IP}:$PORT
30
31  sudo -E nft list ruleset
```

リスト29 [ポートフォワード解除コマンド (net/portfwd_remove.sh)]

```
 1  #!/bin/bash
 2  #
 3  # ポートフォワードの設定を解除するコマンド
 4
 5  if [ "$#" -lt 2 ]; then
 6      echo "Usage: $0 ホストポート番号 ポート番号"
 7      exit 1
 8  fi
 9  HOST_PORT=$1
10  PORT=$2
11
12  echo "Delete Port forward: $HOST_PORT $PORT"
13
14  # 環境情報の読み込み
15  . $MYCON_TOP/MYENV
16
17  # ポートフォワードの設定を削除する
18  NO=`sudo -E nft -a list ruleset | grep $HOST_PORT | grep $PORT | awk '{print $NF}'`
19  sudo -E nft delete rule MYNAT prerouting handle $NO
```